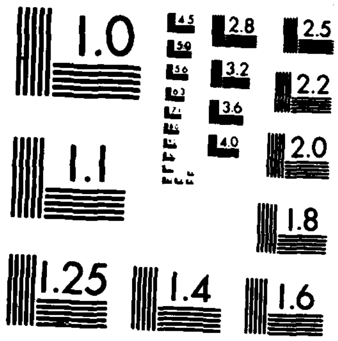
AD-A174 760 MISSION RELIABILITY MODEL PROGRAMMERS GUIDE(U) ANALYTIC SCIENCES CORP READING MA J M MEDINA ET AL DEC 86 AFHRL-TP-86-38 F33615-82-C-0002 1/2

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

AFHRL-TP-86-38

AD-A174 760

AIR FORCE

HUMAN RESOURCES

MISSION RELIABILITY MODEL PROGRAMMERS GUIDE

Joseph M. Medina
Jonathan H. Simonson
Michael H. Veatch

The Analytic Sciences Corporation
One Jacob Way
Reading, Massachusetts 01867

LOGISTICS AND HUMAN FACTORS DIVISION
Wright-Patterson Air Force Base, Ohio 45433-6503

December 1986

Final Technical Paper for Period March 1982 - March 1986

Approved for public release; distribution is unlimited.

LABORATORY

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5601

NOTICE

This paper has been reviewed and is approved for publication.

LEE H. DAYTON, 2Lt, USAF
Contract Monitor

BERTRAM W. CREAM, Technical Director
Logistics and Human Factors Division

DONALD C. TETMEYER, Colonel, USAF
Chief, Logistics and Human Factors Division

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

ADA 174760

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution is unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)
AFHRL-TP-86-38

6a. NAME OF PERFORMING ORGANIZATION
The Analytic Sciences Corporation

6b. OFFICE SYMBOL *(If applicable)*

7a. NAME OF MONITORING ORGANIZATION
Logistics and Human Factors Division

6c. ADDRESS *(City, State, and ZIP Code)*
One Jacob Way
Reading, Massachusetts 01867

7b. ADDRESS *(City, State, and ZIP Code)*
Air Force Human Resources Laboratory
Wright-Patterson Air Force Base, Ohio 45433-6503

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Air Force Human Resources Laboratory

8b. OFFICE SYMBOL *(If applicable)*
HQ AFHRL

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F33615-82-C-0002

8c. ADDRESS *(City, State, and ZIP Code)*
Brooks Air Force Base, Texas 78235-5601

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
62205F	1710	00	26

11. TITLE *(Include Security Classification)*
Mission Reliability Model Programmers Guide

12. PERSONAL AUTHOR(S)
Medina, J.M.; Simonson, J.H.; Veatch, M.H.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Mar 82 TO Mar 86	December 1986	102

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
14	04	
12	01	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
communication; logistics analysis; fault-tolerant avionics; maintainability; identification; mathematical models (Continued)

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The Mission Reliability Model (MIREM) has been developed to evaluate the reliability and sustained operating capability of advanced electronic circuits during the early stages of design. MIREM is applicable to integrated systems that achieve fault tolerance through fault detection, fault isolation, and reconfiguration. The MIREM Programmers Guide discusses the program structure, function of routines, interdependence of subprograms and common blocks, and file usage. The information needed to port the model to other computer systems is also provided. User instructions are discussed in AFHRL-TR-86-35, "Mission Reliability Model Users Guide."

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION

22a. NAME OF RESPONSIBLE INDIVIDUAL
Nancy A. Perrigo, Chief, STINFO Office

22b. TELEPHONE *(Include Area Code)*
(512) 536-3877

22c. OFFICE SYMBOL
AFHRL/TSR

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

Item 18 (Concluded):

mean time between critical failure
mean time between failure
mean time to repair
mission completion success probability
reliability

AFHRL Technical Paper 86-38

December 1986

MISSION RELIABILITY MODEL PROGRAMMERS GUIDE

Joseph M. Medina
Jonathan H. Simonson
Michael H. Veatch

The Analytic Sciences Corporation
One Jacob Way
Reading, Massachusetts 01867

LOGISTICS AND HUMAN FACTORS DIVISION
Wright-Patterson Air Force Base, Ohio 45433-6503

Reviewed by

Lee H. Dayton, 2Lt, USAF
Logistics Systems Branch

Submitted for publication by

Joseph F. Nerad, Major, USAF
Chief, Logistics Systems Branch

This publication is primarily a working paper. It is published solely to document work performed.

SUMMARY

The Mission Reliability Model (MIREM) has been developed to evaluate the reliability and sustained operating capability of advanced electronic circuits during the early stages of development. MIREM is applicable to integrated systems that achieve fault tolerance through dynamic fault detection, fault isolation, and reconfiguration. The model can also be valuable in evaluating designs that employ only dedicated or "hard-wired" redundancy.

The most unique feature of MIREM is its ability to accurately reflect the impact of reconfigurable, competing functions on system reliability. The user defines the resources necessary to support a required function, e.g., Global Positioning System (GPS), and the model will compute the probability of losing that functional capability over a certain operating time. A critical failure occurs when there are not a sufficient number of working resources to support a specified function. As an analytic model, MIREM determines a value for Mean Time Between Critical Failure, Mission Completion Success Probability, and Failure Resiliency.

The MIREM Programmers Guide addresses the model's program structure, function of routines, interdependence of subprograms and common blocks, and file usage. The information needed to port the model to other computer systems is also provided.

Keywords: avionics, mission reliability, mathematical models.

PREFACE

The Mission Reliability Model (MIREM) has been developed to evaluate the reliability and operating capability of advanced avionics in the conceptual stages of design. The model is applicable to systems, such as the Integrated Communication, Navigation, and Identification Avionics (ICNIA) architectures, that achieve fault tolerance through dynamic fault detection and reconfiguration.

This guide provides documentation on MIREM for program maintenance. It is consistent with the software release dated 17 March 1986. Programming topics such as the program structure, function of routines, interdependence of subprograms and common blocks, and file usage are addressed. The information needed to port the programs to other computer programs is also provided. User instructions for this model are provided in AFHRL-TR-86-35, "Mission Reliability Model Users Guide."

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The Mission Reliability Model (MIREM) is a fault-tolerant system reliability program developed to evaluate the mission reliability and sustained operating capability of advanced electronics systems during the early development phase. These systems contain integration, redundancy, and dynamic reconfigurability (self-repair) as part of their fault-tolerant design. Typical reliability analyses that can be conducted using MIREM include:

1. Evaluation of mission reliability for alternative mission scenarios.
2. Determination of the additional operating time without repair that can be achieved due to fault tolerance.
3. Identification of the parts within a system that are contributing significantly to mission failures.
4. Identification of design improvements that offer a large payoff in mission reliability.
5. Comparison of integrated, fault-tolerant systems with conventional discrete systems in terms of mission reliability.

These analysis capabilities are provided by a new mathematical model constructed to assess a broad class of fault-tolerant structures.

The DATAIN, MIREM, and MPLOT computer programs implement the MIREM model. Figure 1 displays the relationship among these programs. The DATAIN program performs the data entry function using online, user-friendly screens to create or update architecture files and scenario files. The MIREM program reads these files to perform computations, prepares reports dealing with fault-tolerant system reliability, and creates a plot file containing the selected plots. Finally, MPLOT reads the plot files and produces plots selected by the user.

This document serves as a programmer's guide to the DATAIN, MIREM, and MPLOT programs to help with program maintenance. Hence, this document deals with programming topics such as

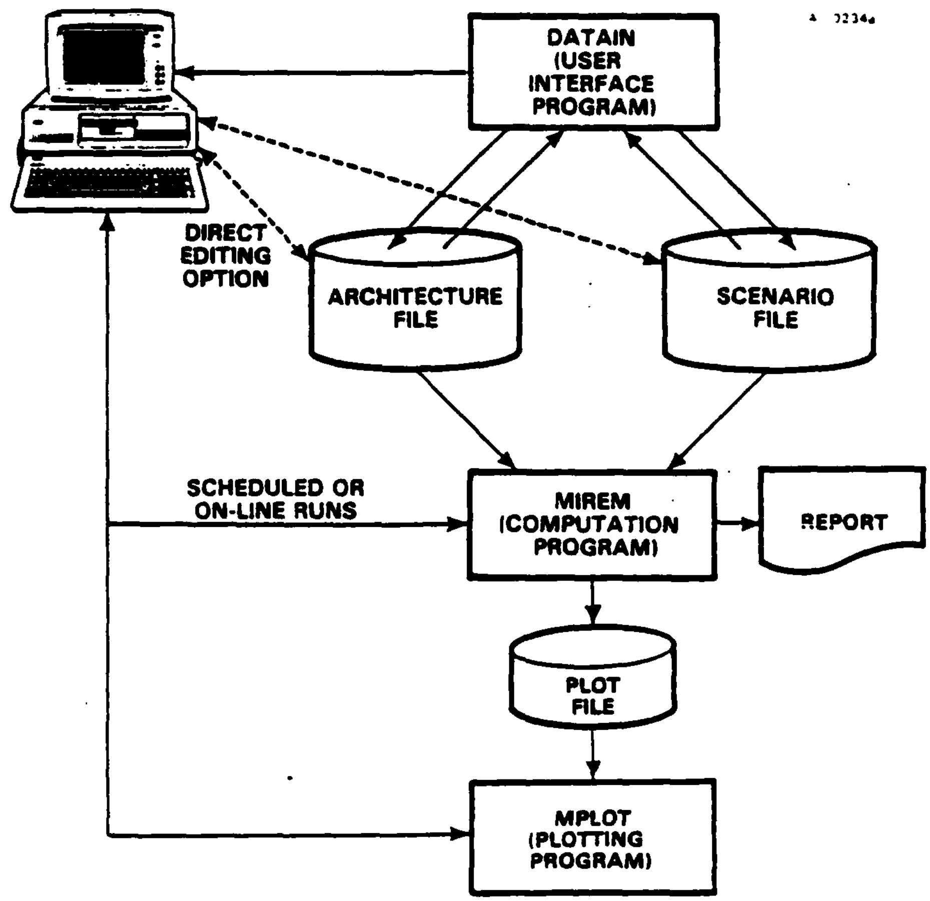


Figure 1. MIREM Program Overview.

program structure, the purpose of the routines comprising the program, the interdependence of subprograms and common blocks, and file usage. Chapter 2 covers these topics for the DATAIN program; Chapter 3, for the MIREM program; and Chapter 4, for the MPLOT program. Additional documentation such as detailed processing descriptions, input and output parameter descriptions, and COMMON block parameter descriptions may be found within comment statements of the source code of the various subprograms.

These programs use the FORTRAN 77 computer language in order to enhance portability of the code to a variety of computer installations. Although the programs use the ANSI standard coding rules, a few portability considerations require further explanation. Chapter 5 describes these portability details.

Appendix A presents sample files, Appendix B lists sample reports generated by the model, and Appendix C shows some sample plots generated by MPLOT.

A second document - AFHRL-TR-86-35, "Mission Reliability Model Users Guide" - contains much information that this document complements but does not reproduce. In particular, the Users Guide includes the history of the MIREM model, the algorithms used in MIREM, the structure of the scenario and architecture files, sample MIREM reports, and a guide to using the program. Because of the complementary nature of this Programmers Guide to the Users Guide, the reader of this manual should be familiar with the contents of the Users Guide.

2. DATAIN PROGRAM

2.1 General Description of DATAIN

The DATAIN program allows the user to interactively prepare data files that are read by the program MIREM. This capability is particularly useful on architecture files since these files can be large and complex. The user may, however, opt to create and modify the files using the computer system's editor.

The DATAIN program consists of 98 FORTRAN 77 routines - a main routine (DATAIN), 89 subroutines, and eight functions. The program also calls the DATE and TIME VAX/VMS intrinsic subroutines and the LIB$ERASE_PAGE VAX run-time function.

During the entry of data through DATAIN, many checks are performed that are also contained in the MIREM program. DATAIN implements the data entry/checking in a series of user-friendly screens controlled by the user.

Section 2.2 describes the basic structure of the DATAIN program. A brief description of the 98 routines is provided in Section 2.3. Section 2.4 discusses the subprogram calling structure and common block usage. Finally, the files used by the program are described briefly in Section 2.5.

2.2 DATAIN Program Structure

This section describes the high-level structure of the DATAIN program. Section 2.2.1 discusses the main program and how it controls the various functions implemented by DATAIN. Section 2.2.2 describes the processing that controls the reading of architecture and scenario files. Section 2.2.3 discusses the routines that control the modification of architecture data, and Section 2.2.4 performs the same function for scenario data.

2.2.1 The DATAIN Main Program

The main routine of DATAIN controls the implementation of the four principal functions that the program performs:

1. Creation of an Architecture File.

2. Modification of an Architecture File.

3. Creation of a Scenario File.

4. Modification of a Scenario File.

Figure 2 shows the tree diagram for the main routine illustrating the principal subroutines and their relationship to the main program. A flowchart that describes the processing control of the main routine is shown in Figure 3.

The subroutine INIT performs all the initialization for DATAIN, which occurs only once. In addition, it produces the Introduction Screen, which identifies the program, and also the Control Keywords Screen if the user requests to see it from the Introduction Screen.

After calling the subroutine INIT, the program enters the main processing loop, which consists of a menu selection screen from which the user selects one of the four functions listed above, followed by the subroutines needed to execute the functions. There are four subroutines used to build and control the Dialogue Selection Menu Screen containing the four functions. They are: UFINIT, UFCSET, UFMENU, and COMAND. UFINIT initializes the screen-generation software for a new screen. UFCSET defines the set of valid commands for a screen. UFMENU displays a menu selection screen and returns the user's selection(s) and a code for some commands that may have been exercised. COMAND is used in the event that the 'Help' or 'Quit' commands are requested to display the appropriate screen and return the command code entered by the user. These four subroutines are utility subroutines that are used frequently throughout the program for screen generation.

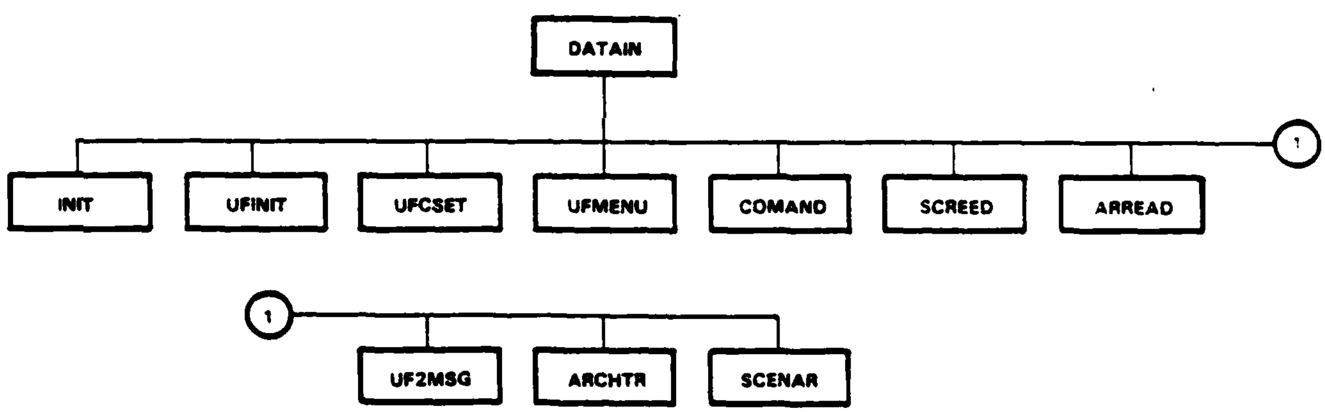


Figure 2. DATAIN Tree Diagram - Main Program.

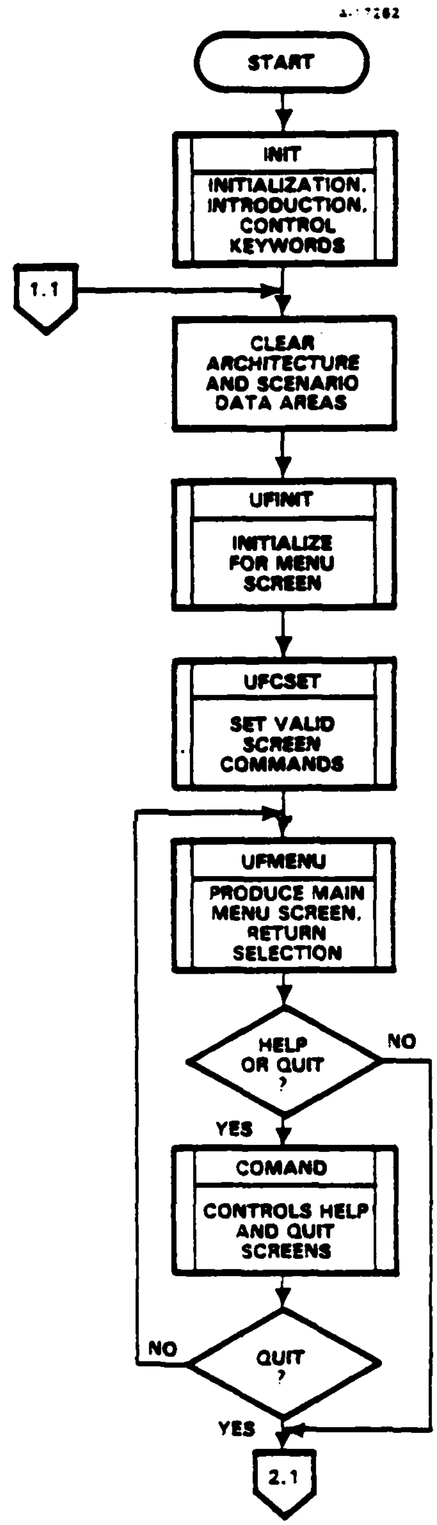


Figure 3. DATAIN Flowchart - Main Program.

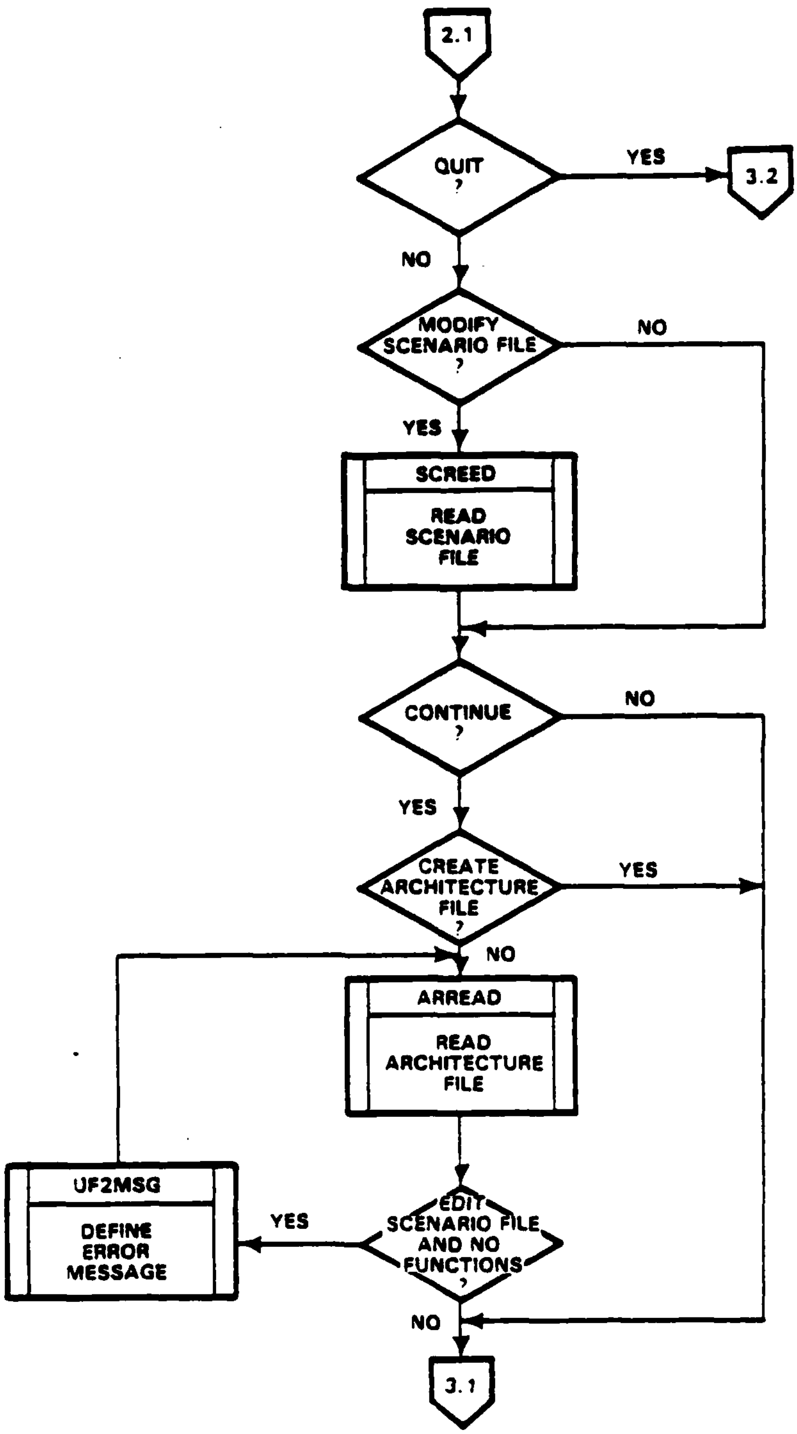


Figure 3. (Continued)

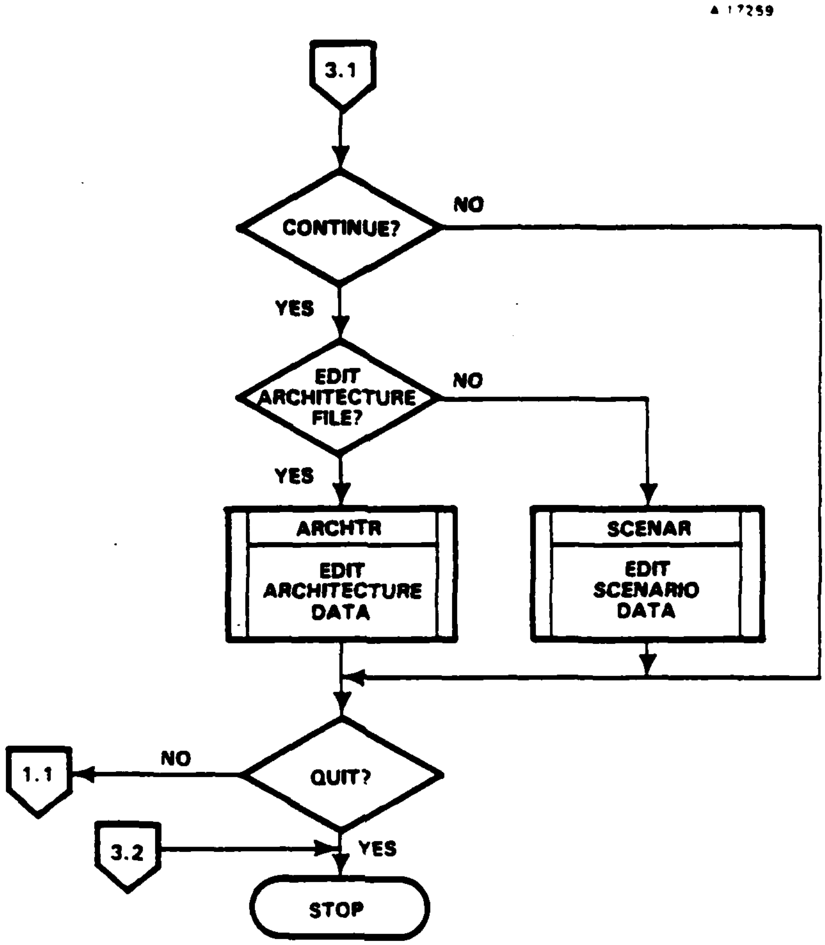


Figure 3. (Concluded)

If the user selects 'Create an Architecture File' from the Dialogue Selection Menu, DATAIN calls the subroutine ARCHTR, which controls the editing and saving of architecture data. If the user selects 'Modify an Architecture File' from the Dialogue Selection Menu, DATAIN calls the subroutine ARREAD, which reads the architecture file to be modified. After the architecture file named by the user has been successfully read, ARCHTR is then called to edit and save the architecture data.

If the user selects 'Create a Scenario File' from the Dialogue Selection Menu, DATAIN calls ARREAD, to read the function list from the architecture file that the scenario data are to reference. An architecture file with a non-empty function list must be provided to be able to edit any scenario file. After the architecture file has been successfully read, DATAIN calls the routine SCENAR, which controls the editing and saving of scenario data. Finally, if the user selects 'Modify a Scenario File' from the Dialogue Selection Menu, DATAIN calls SCREED, to

read the scenario file to be modified. After the scenario file named by the user has been read, ARREAD is called to read the function list from the architecture file. After this, SCENAR is called to complete the editing and save the scenario data.

2.2.2 Reading the Input Files

The routines SCREED and ARREAD control the reading of the scenario and architecture files, respectively. Examples of these two file types are found in Appendix A.

Scenario files are read by the subroutine SCREED. Figure 4 shows the tree diagram of the subroutines called by SCREED. The routine GETFIL displays a screen which asks the user for the name of the scenario file to be read. RUNID, HARDWARE, COMPUTE, PLOT, TIME, REPSEQUENCE, PRINT, and MISSION cards are interpreted by the routines SCRNI, SCHRD, SCCMP, SCPLT, SCTIMS, SCREPS, SCPRT, and SCMIS, respectively. SIMULTANEOUS and QUICK cards are both interpreted by the subroutine SCYORN. The SCALE card is interpreted by SCREAL. In addition, SCPRT uses the routine SCYORN to process the PRINT cards. The routines UF1PAR and UF2MSG are utility routines which are used to do string parsing and error message generation, respectively.

Architecture files are read by the subroutine ARREAD. The tree diagram of the subroutines called from ARREAD is shown in Figure 5. The routine GETFIL is used to get the name of the architecture file from the user. ARFCN, ARLRU, ARRES, ARCHI, and ARPOL are called to interpret FUNCTION, LRU, RESOURCE, CHAIN, and POOL cards, respectively. The routines UF1PAR and UF2MSG are also used for the same purposes as described above.

2.2.3 Editing Architecture Data

The routine ARCHTR controls the editing of architecture data by the user. This is done whenever the user selects the Create or Modify Architecture File options from the Dialogue Selection Menu screen in the main routine of DATAIN. Figure 6 shows the tree diagram of the major subroutines which take part in the editing of architecture data. The flow of control executed by ARCHTR is shown in the flowchart in Figure 7. The main processing within ARCHTR is a loop which begins with the Architecture File Menu Screen. This screen allows the user to select the type of data within the file to be modified, or to save the data in a new file when ready. Following the user's selection, the appropriate activity is initiated. When the activity has been completed, the Architecture File Menu Screen is again shown, and the process is repeated.

The Architecture File Menu Screen is built and controlled by the routines UFINIT, UFCSET, UFMENU, and COMAND. These routines are described briefly in Section 2.2.1.

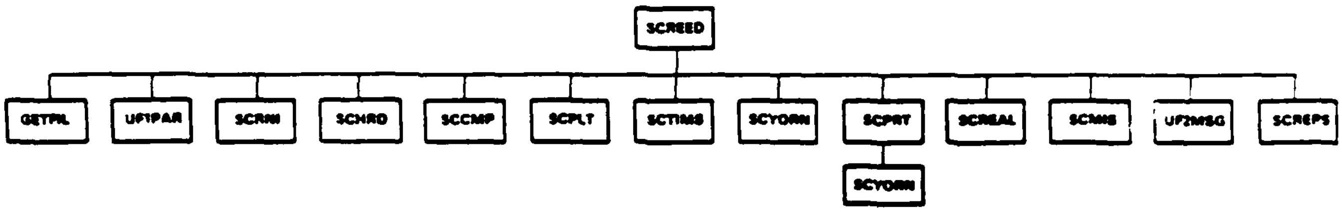


Figure 4. SCREED Tree Diagram - Read Scenario File.

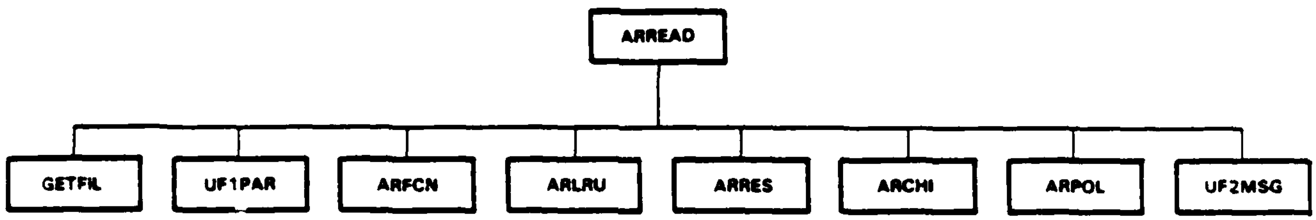


Figure 5. ARREAD Tree Diagram - Read Architecture File.

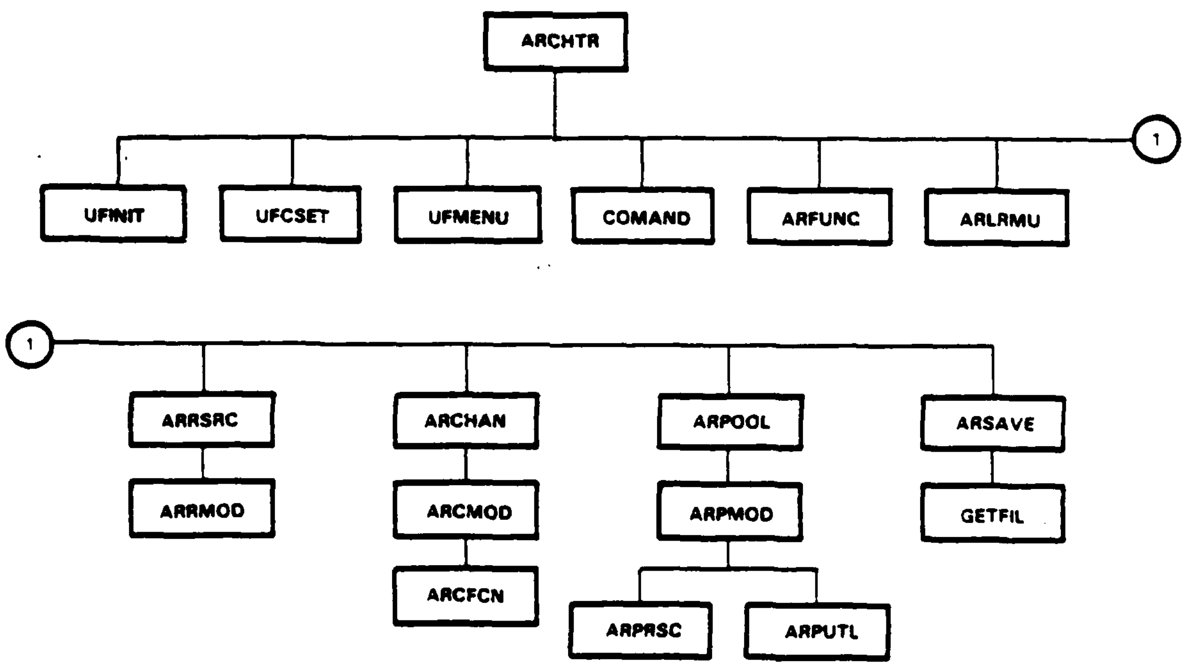


Figure 6. ARCHTR Tree Diagram - Edit Architecture Data.

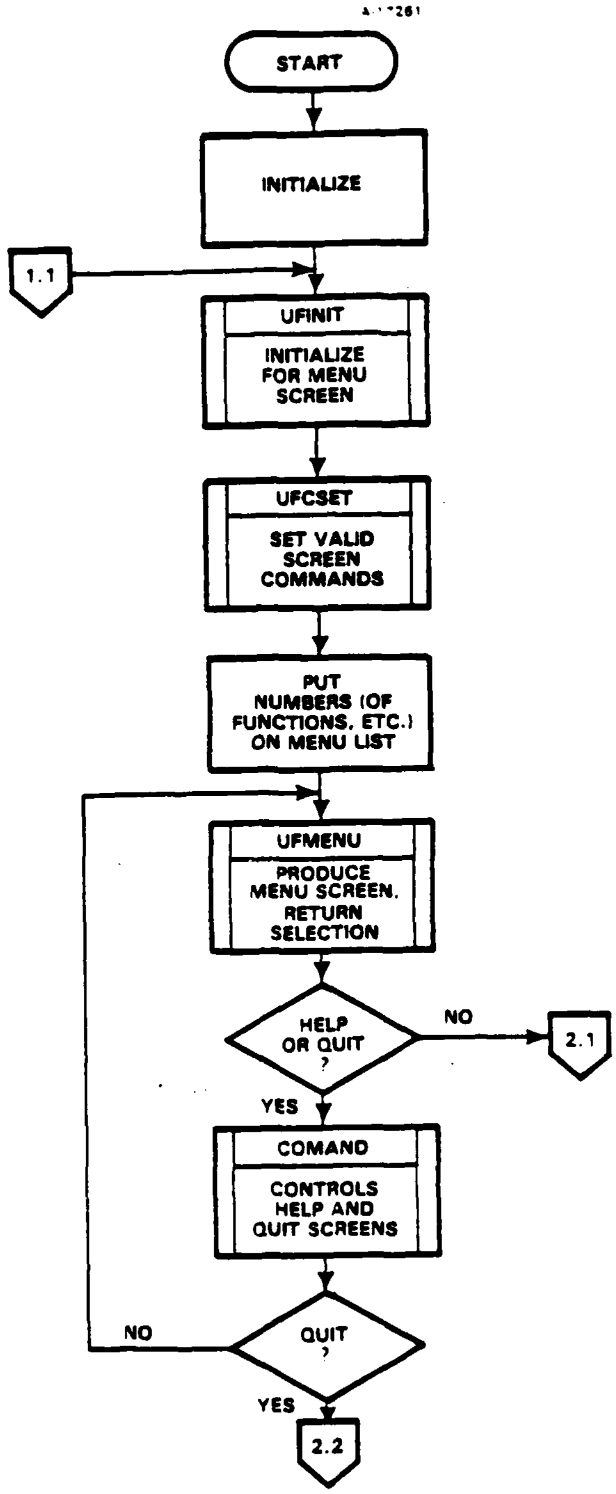


Figure 7. DATAIN Flowchart - ARCHTR.

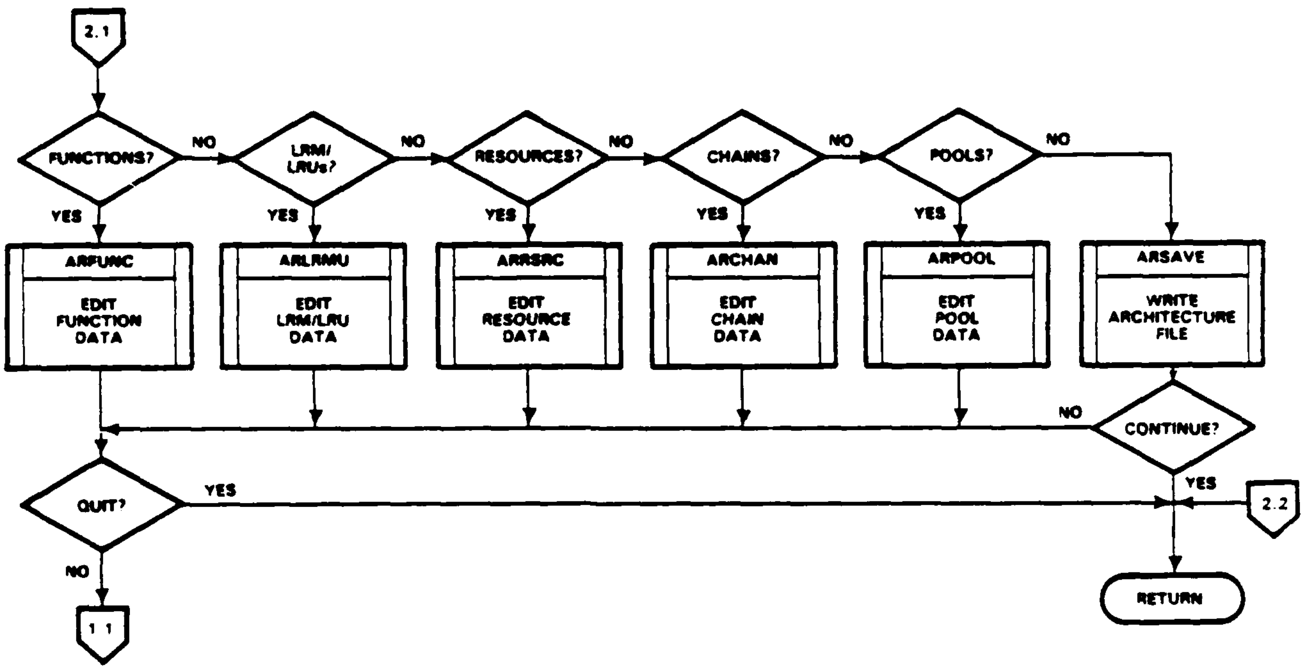


Figure 7. (Concluded)

When the user selects 'Functions' on the Architecture File Menu Screen, ARCHTR calls the routine ARFUNC to edit the function data. This routine produces a data-entry screen on which the user can add, change, or delete functions. When the user is finished with this process, the Architecture File Menu Screen is shown again.

When 'LRM/LRUs' is selected from the Architecture File Menu Screen, ARLRMU is called by ARCHTR to edit the LRM/LRU data. This routine operates in a manner similar to ARFUNC.

When the user selects 'Resources', ARCHTR calls ARRSRC. ARRSRC controls the adding, changing, and deletion of resources in the architecture file. If there are resources in the file when ARRSRC is entered, ARRSRC displays the list of those resources. The user can select resources to be changed or deleted and also indicate whether or not resources are to be added. If resources are being changed or added, ARRSRC calls ARRMOD for each resource and the user enters the data. When this process is complete, the user is shown the new resource list. This process is repeated until the user makes no changes, deletions, or additions on the resource list screen. If the user selects 'Resources' and there are no resources in the architecture data, ARRSRC goes directly into the add resource mode and bypasses the resource list screen.

When 'Chains' is selected from the Architecture File Menu Screen, ARCHAN is called by ARCHTR to control the editing of chain data. This dialogue is similar to the resource dialogue described above. ARCHAN displays the list of chains (if there are any) to which the user can make changes, deletions, or additions. When a chain is being changed or added, ARCHAN calls ARCMOD to control the process. For each chain being changed or added, there is a chain data entry screen displayed by ARCMOD and one or two chain function screens (depending on whether the primary chain is paired with a secondary chain or not). The chain function screen allows the user to select, from the list of functions in the architecture data, the functions that are in the particular chain. This screen is implemented by the routine ARCFCN and called from ARCMOD. Here, also, the add chain mode is entered directly if there are no chains in the architecture data when 'Chains' is selected.

When the user selects 'Pools', ARCHTR calls the routine ARPOOL. ARPOOL controls the editing of pool data in the architecture file and works in a manner similar to the chain dialogue described above. ARPOOL displays the list of pools in the architecture data when 'Pools' is selected. The user can choose to change, replicate, delete, or add pools. The replicate feature allows the user to add a pool and copy all data describing the pool from one which already exists. This is particularly useful for pools in secondary chains of a chain pair. ARPMOD controls the changing or addition of pools. This process is done in three screens. The pool data entry screen is displayed by ARPMOD itself. In addition, ARPMOD calls ARPRSC to allow the user to enter the resources that are in the pool, and ARPUTL for entering the pool utilization rates corresponding to the functions in the architecture data. The add pool mode is entered directly whenever 'Pools' is selected and there are no pools in the architecture data.

Finally, ARSAVE is called by ARCHTR when 'Save' is selected by the user in the Architecture File Menu screen. This routine stores the architecture data in a file for the user. ARSAVE calls GETFIL to allow the user to enter a name for the architecture file. In addition, ARSAVE makes additional checks on the data and may produce a report of inconsistencies found in the data. The checks that are performed are primarily concerned with pool data in chain pairs. If ARSAVE completes the writing of the file with no inconsistencies detected, ARCHTR returns the user to the Dialogue Selection Menu; if inconsistencies are detected, the user is returned to the Architecture File Menu screen.

2.2.4 Editing Scenario Data

The routine SCENAR controls the editing of scenario data by the user. This is done when the user selects the Create or Modify Scenario File options from the Dialogue Selection Menu screen in the main routine of DATAIN. The tree diagram of the major subroutines which take part in the editing of scenario data is shown in Figure 8. Figure 9 is a flowchart of the flow of control as executed by SCENAR. The scenario data editing is not controlled by a menu screen in a loop as the architecture editing is since the scenario data are considerably less complex and less voluminous. Instead, SCENAR steps through the various screens in a linear fashion until the process is completed by storing the data in a file. Through use of the 'Back' command, the user may return to a previous screen and make corrections.

SCENAR calls the routine SCRID to allow the user to enter the run identification for the scenario data. SCENAR calls the routine SCTSEL to ask the user to enter operating times. The routine SCCSEL is called by SCENAR so that the user may enter any computation selections for the run. Similarly, SCENAR calls SCPSEL so that the user may enter plot selections for the run. SCENAR calls the routine SCPARM to allow the user to enter the run parameters (such as PRINT, SCALE, SIMULTANEOUS, etc).

The routine SCPHAS is called by SCENAR to control the editing of mission phase data by the user. This process works similarly to the editing of chain data in the architecture data processing. SCPHAS displays the list of mission phases (if any exist) to which the user can make changes, deletions,

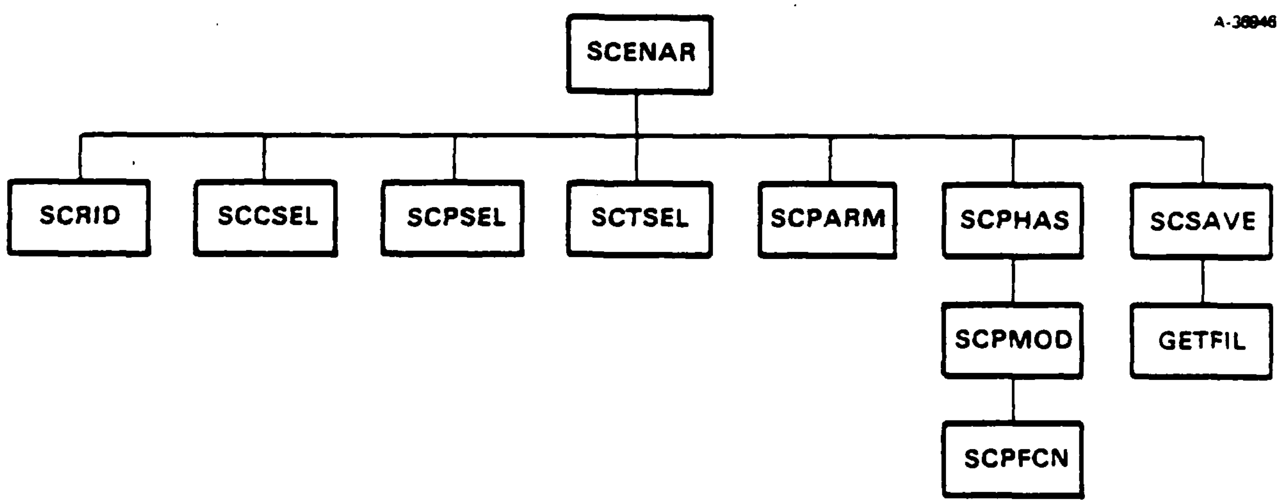


Figure 8. SCENAR Tree Diagram - Edit Scenario Data.

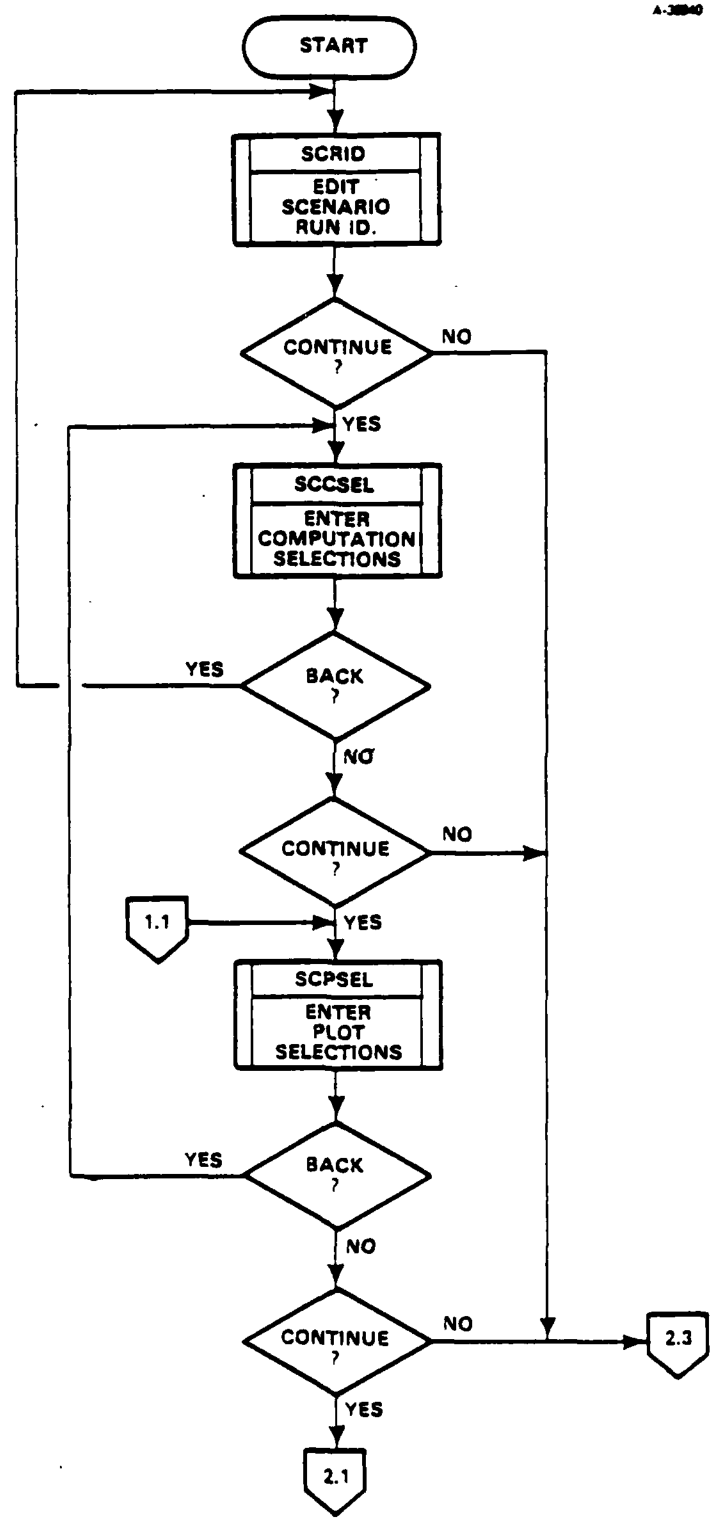


Figure 9. DATAIN Flowchart - SCENAR.

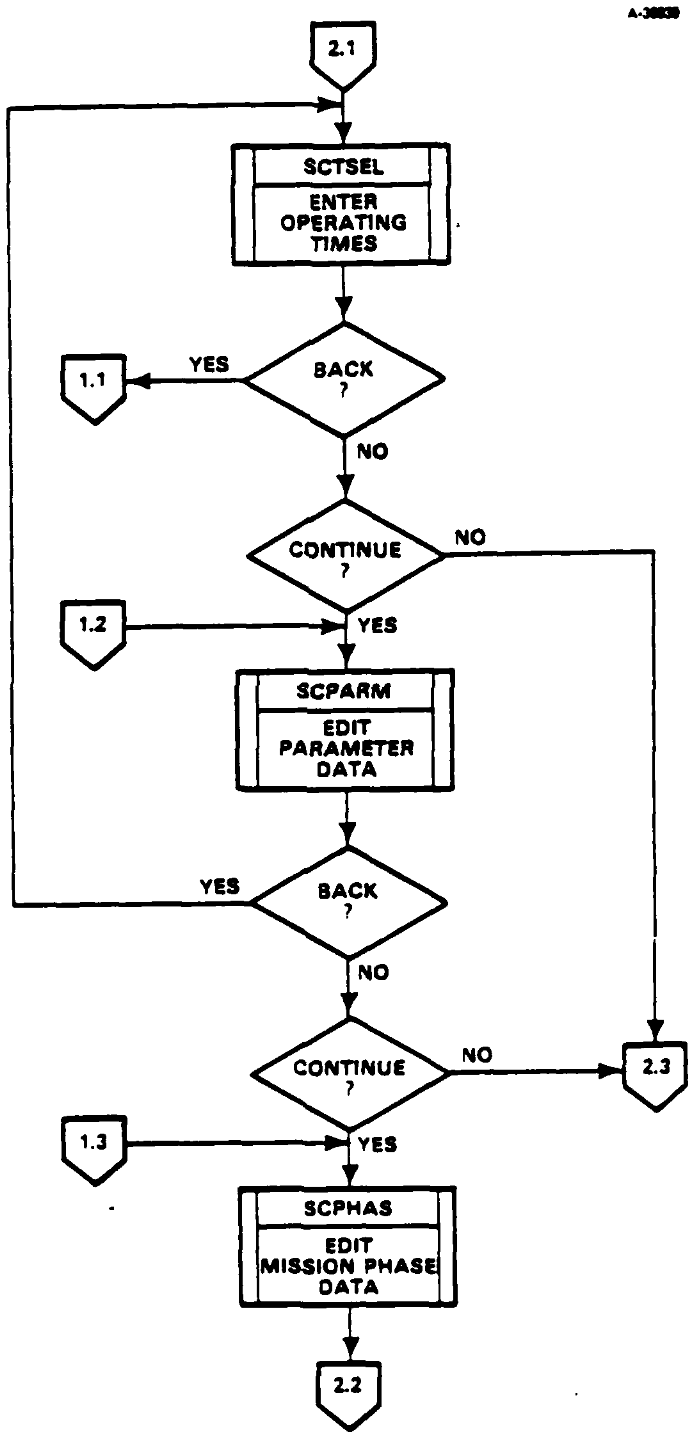


Figure 9. (Continued)

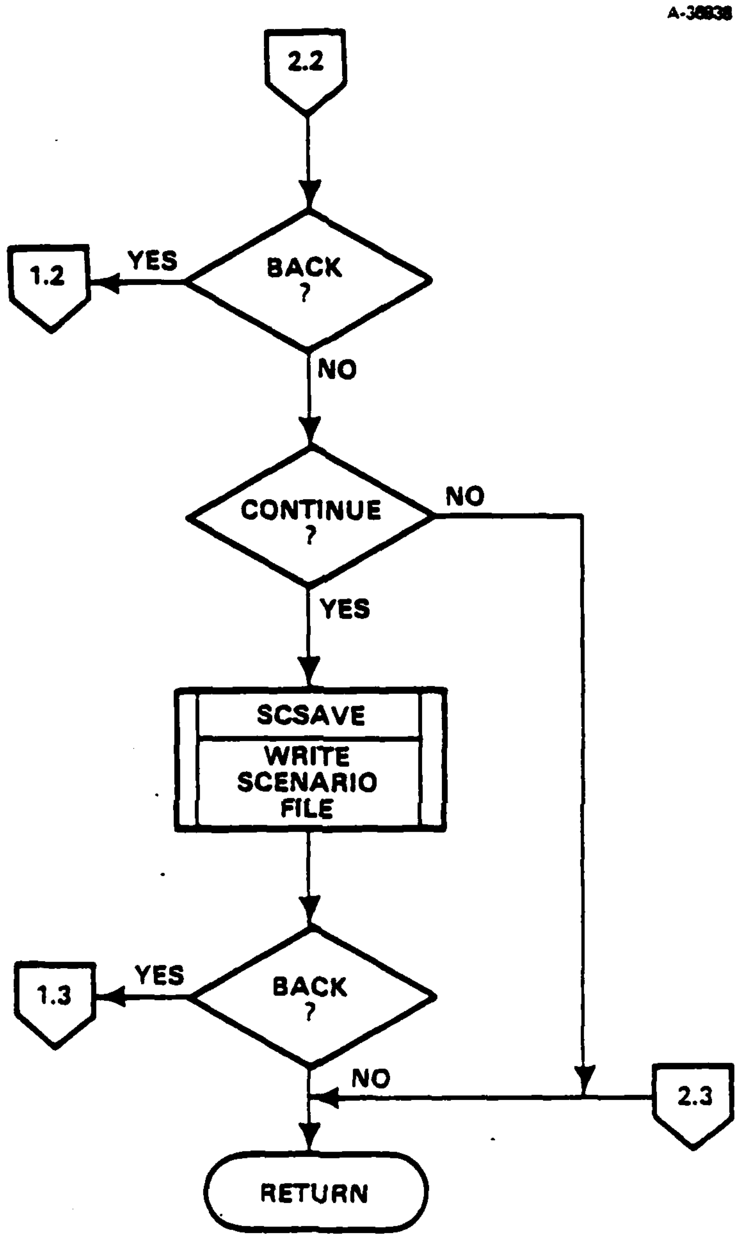


Figure 9. (Concluded)

or additions. Changes or additions to mission phase data are controlled by the routine SCPMOD, called from SCPHAS, and are implemented in two screens. SCPMOD displays the mission phase data entry screen and calls the routine SCPFCN to allow the user to designate the critical functions (from the architecture data) that are in the phase. The add phase mode is entered directly when SCPHAS is called with no mission phases in the scenario data. The user is returned to the mission phase list

screen after each set of modifications, until no changes, deletions, or additions are made on that screen.

Following the completion of editing on the mission phase data, SCENAR calls the routine SCSAVE to allow the user to store the scenario data in a file. This routine works the same as the routine ARSAVE described in Section 2.2.3 except that no additional data checks are made. GETFIL is called for the user to enter a name for the scenario file being created. After successful completion of storing the scenario file, SCSAVE returns the user to the Dialogue Selection Menu.

2.3 DATAIN Subroutine Descriptions

The 98 subprograms used in the DATAIN program are briefly described in Table 1. They are presented in alphabetic order, with the exception of the main routine, DATAIN, which appears first.

2.4 DATAIN Subprogram Interdependencies and Common Block Usage

This section identifies the dependencies between the subprograms in DATAIN in terms of transfer of control and the usage of COMMON storage. Table 2 lists, for each routine, the subprograms it calls, routines that call it, and COMMON blocks used. The description of the various parameters in the COMMON blocks are found in the routine header comments for the main program, DATAIN.

Table 1. DATAIN Routine Descriptions

Routine	Description
DATAIN	Produces Dialogue Selection Menu, reads appropriate files, and edits selected data.
ARCFCN	Controls the editing of functions in a chain by the user.
ARCHAN	Controls the entry of chains by the user.
ARCHI	Interprets architecture chain records.
ARCHTR	Controls the edit processing of architecture files.
ARCMOD	Controls the editing of a chain by the user.
ARCONV	Converts a character vector to upper case and tests for invalid characters.
ARFCN	Interprets architecture function records.
ARFIND	Finds the position of a given number in a list.
ARFUNC	Controls the entry of functions by the user.
ARLRMU	Controls the entry of LRM/LRU s by the user.
ARLRU	Interprets architecture LRM/LRU records.

Table 1. (Continued)

Routine	Description
ARNEXT	Finds the next available place for a number in a list.
ARPMOD	Controls the editing of a pool by the user.
ARPOL	Interprets architecture pool records.
ARPOOL	Controls the entry of pools by the user.
ARPRSC	Controls the editing of pool resources by the user.
ARPSRT	Sorts the list of pools by chain then pool number.
ARPUTL	Controls the editing of pool utilizations by the user.
ARREAD	Controls reading of the architecture file.
ARRES	Interprets architecture resource records.
ARRMOD	Controls the editing of a resource by the user.
ARRSRC	Controls the entry of resources by the user.
ARSAVE	Saves the architecture data in a file.
ARSORT	Sorts a list of positive numbers.
COMAND	Controls execution of DATAIN commands.
GETFIL	Reads a file name entered by the user and opens the file.
INIT	Controls the initialization processing for DATAIN.
SCCMP	Interprets scenario computation selection records.
SCCSEL	Controls the entry of the computation selections by the user.
SCENAR	Controls the edit processing of scenario files.
SCHRD	Interprets scenario architecture file name records.
SCMIS	Interprets scenario mission records.
SCPARM	Controls the entry of scenario parameters by the user.
SCPFCN	Controls the editing of critical functions in a mission phase by the user.
SCPHAS	Controls the entry of mission phases by the user.
SCPLT	Interprets scenario plot selection records.
SCPMOD	Controls the editing of a mission phase by the user.
SCPRT	Interprets scenario print control records.
SCPSEL	Controls the entry of the plot selections by the user.
SCREAL	Interprets scenario real number records.
SCREED	Controls reading of the scenario file.
SCREPS	Interprets scenario repair sequence records.
SCRID	Controls the entry of the run identifier by the user.
SCRNI	Interprets scenario run identification records.
SCSAVE	Saves the scenario data in a file.
SCTEST	Tests and translates character abbreviations to unabbreviated form.
SCTIMS	Interprets scenario operating times records.
SCTSEL	Controls the entry of the operating time by the user.
SCYORN	Interprets scenario yes or no option records.
UFCBAD	Invalid command processing routine.
UFCDEF	Command definition routine.
UFCEXE	Command execution routine.
UFCFND	Command finding routine.
UFCHBL	'HELP' screen building routine.

Table 1. (Concluded)

Routine	Description
UFCHLP	'HELP' command execution routine.
UFCSET	Command setting routine.
UFECST	Data entry character setting routine.
UFEDSP	Data entry display and response routine.
UFEIST	Data entry integer setting routine.
UFELST	Data entry listing routine.
UFENTR	Controls the display and validates responses for data entry screens.
UFERST	Data entry real setting routine.
UFESET	Data entry variable setting routine.
UFINIT	Screen initialization routine.
UFIT	Initialization routine.
UFMDEF	Menu default selection setting routine.
UFMDSP	Menu display and response routine.
UFMENU	Controls the display and validates responses for menus.
UFMFMT	Format setting routine for menu-type screens.
UFMLST	Menu selection listing routine.
UFMRET	Menu selection returning routine.
UFMSEL	Selection number setting routine for menu-type screens.
UFREPT	Controls the display and validates responses for reports.
UF1CEL	Ceiling function for integer division.
UF1CLR	Screen area clearing routine.
UF1DBL	Routine to strip blank characters.
UF1ERS	CRT screen erase routine.
UF1FMT	Numeric format generation routine.
UF1GET	Read line routine.
UF1ICK	Integer range checking routine.
UF1IND	Integer-to-character conversion preparation.
UF1INT	Character-to-integer conversion routine.
UF1LCL	Routine to determine list column positions.
UF1LST	Routine to find the first line for a list.
UF1NCL	Routine to compute the number of list columns.
UF1OPN	File opening routine.
UF1PAR	Token parsing routine.
UF1PUT	Write line routine.
UF1RCK	Real range checking routine.
UF1REL	Character-to-real conversion routine.
UF1ROP	Close and reopen the Terminal File for end-of-file processing.
UF1UPR	Lower-to-upper case conversion routine.
UF1WID	Screen width setting routine.
UF2MSG	Error message routine.
UF2PAG	Page display routine.
UF2TXT	Screen text definition routine.
UNQUOT	Routine to strip QUOTES from quote-delimited strings.

Table 2. DATAIN Subprogram Interdependencies and Common Block Usage

Routine	Routines Called	Calling Routines	Common Blocks
DATAIN	INIT, UFINIT, UFCSET, UFMENU, COMAND, SCREED, ARREAD, UF2MSG, ARCHTR, SCENAR	None (Main Program)	ARCHN, ARCHC, SCENN, SCENC, UFITN, UFITC
ARCFCN	UF1IND, UF1CEL, UFINIT, UFCSET, UFMFMT, UFMSEL, UFMENU, COMAND, UF2MSG	ARCMOD	ARCHN, ARCHC, UFITN, UFITC, UFPCI
ARCHAN	ARSORT, UFINIT, UFCSET, UFENTR, COMAND, ARCONV, UF2MSG, ARCMOD, ARNEXT	ARCHTR	ARCHN, ARCHC, UFICI, UFICC, UFITN, UFITC, UFPCI
ARCHI	UF1PAR, UF1FMT, UF1INT, UNQUOT, ARFIND, ARNEXT, UF2MSG	ARREAD	ARCHN, ARCHC
ARCHTR	UFINIT, UFCSET, UFMENU, COMAND, ARFUNC, ARLRMU, ARRSRC, ARCHAN, ARPOOL, ARSAVE	DATAIN	ARCHN, UFITN, UFITC
ARCMOD	UFINIT, UFCSET, UFENTR, COMAND, UF2MSG, ARCFCN	ARCHAN	ARCHN, ARCHC, UFICI, UFICC, UFITN, UFITC
ARCONV	UF1UPR	ARCHAN, ARPOOL, ARRSRC, SCPHAS	None
ARFCN	UF1PAR, UNQUOT, UF2MSG	ARREAD	ARCHN, ARCHC
ARFIND	None	ARCHI, ARPMOD, ARPOL, ARPOOL, ARPRSC, ARSAVE	None
ARFUNC	UFINIT, UFCSET, UFMFMT, UFENTR, COMAND, UF2MSG	ARCHTR	ARCHN, ARCHC, UFICI, UFICC, UFITN, UFITC
ARLRMU	UFINIT, UFCSET, UFMFMT, UFENTR, COMAND, UF2MSG	ARCHTR	ARCHN, ARCHC, UFICI, UFICC, UFITN, UFITC
ARLRU	UF1PAR, UNQUOT, UF2MSG	ARREAD	ARCHN, ARCHC

Table 2. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
ARNEXT	None	ARCHAN, ARCHI, ARPOOL, ARRSRC, ARSAVE, SCMIS, SCPHAS, SCSAVE	None
ARPMOD	UFINIT, UFCSET, UFENTR, COMAND, ARFIND, UF2MSG, UF1UPR, ARPRSC, ARPUTL	ARPOOL	ARCHN, ARCHC, UFICI, UFICC, UFITN, UFITC
ARPOL	UF1PAR, UF1FMT, UF1INT, ARFIND, UNQUOT, UF1REL, UF2MSG	ARREAD	ARCHN, ARCHC
ARPOOL	ARPSRT, UF1IND, UFINIT, UFCSET, UFENTR, COMAND, ARCONV, UF2MSG, ARPMOD, ARNEXT, ARFIND	ARCHTR	ARCHN, ARCHC, UFICI, UFICC, UFITN, UFITC, UFPCI
ARPRSC	UF1IND, ARFIND, UFINIT, UFCSET, UFENTR, COMAND, UF2MSG	ARPMOD	ARCHN, ARCHC, UFICI, UFICC, UFITN, UFITC
ARPSRT	None	ARPOOL, ARSAVE	None
ARPUTL	UF1IND, UF1CEL, UFINIT, UFCSET, UFMFMT, UFENTR, COMAND	ARPMOD	ARCHN, ARCHC, UFICI, UFICR, UFICC, UFITN, UFITC, UFPCI
ARREAD	GETFIL, UF1PAR, ARFCN, ARLRU, ARRES, ARCHI, ARPOL, UF2MSG	DATAIN	ARCHC, UFITN
ARRES	UF1PAR, UF1FMT, UF1INT, UNQUOT, UF2MSG	ARREAD	ARCHN, ARCHC
ARRMOD	UFINIT, UFCSET, UFENTR, COMAND, UF1UPR, UF2MSG	ARRSRC	ARCHN, ARCHC, UFICI, UFICC, UFITN, UFITC
ARRSRC	ARSORT, UFINIT, UFCSET, UFENTR, COMAND, ARCONV, UF2MSG, ARRMOD, ARNEXT	ARCHTR	ARCHN, ARCHC, UFICI, UFICC, UFITN, UFITC, UFPCI

Table 2. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
ARSAVE	GETFIL, ARSORT, ARNEXT, ARPSRT, UFINIT, UFCSET, ARFIND, UF2TXT, UFREPT, COMAND	ARCHTR	ARCHN, ARCHC, UFITN, UFITC, UFPCI, UFPCC
ARSORT	None	ARCHAN, ARRSRC, ARSAVE, SCPHAS, SCSAVE	None
COMAND	UFCHLP	DATAIN, ARCFCN, ARCHAN, ARCHTR, ARCMOD, ARFUNC, ARLRMU, ARPMOD, ARPOOL, ARPRSC, ARPUTL, ARRMOD, ARRSRC, ARSAVE, GETFIL, INIT, SCCSEL, SCPARM, SCPFCN, SCPHAS, SCPMOD, SCRID	UFITN
GETFIL	UF1DBL, UFINIT, UFCSET, UFENTR, COMAND, UF1UPR, UF1OPN, UF2MSG	ARREAD, ARSAVE, SCREED, SCSAVE	UFICI, UFICC, UFITN, UFITC
INIT	UFIT, UFMFMT, UFCDEF, UFCSET, UFINIT, UF2TXT, UFREPT, COMAND	DATAIN	UFICI, UFICR, UFICC, UFITN, UFITC
SCCMP	UF1PAR, UNQUOT, UF2MSG	SCREED	SCENN
SCCSEL	UFINIT, UFCSET, UFMSEL, UFMENU, COMAND	SCENAR	SCENN, UFITN, UFITC
SCENAR	SCRID, SCCSEL, SCPARM, SCPHAS, SCSAVE	DATAIN	UFITN
SCHRD	UF1PAR, UNQUOT, UF2MSG	SCREED	ARCHN
SCMIS	UF1PAR, UF1FMT, UF1INT, UF1REL, UNQUOT, ARNEXT, UF2MSG	SCREED	SCENN, SCENC
SCPARM	UFINIT, UFCSET, UFENTR, COMAND, SCTEST, UF2MSG	SCENAR	SCENN, UFICR, UFICC, UFITN, UFITC
SCPFCN	UF1IND, UF1CEL, UFINIT, UFCSET, UFMFMT, UFMSEL, UFMENU, COMAND	SCPMOD	ARCHN, ARCHC, SCENN, UFITN, UFITC, UFPCI

Table 2. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
SCPHAS	ARSORT, UFINIT, UFCSET, ARNEXT, UFENTR, COMAND, ARCONV, UF2MSG, SCPMOD	SCENAR	SCENN, SCENC, UFICI, UFICC, UFITN, UFITC, UFPCI
SCPLT	UF1PAR, UNQUOT, UF2MSG	SCREEN	SCENN, SCENC
SCPMOD	UFINIT, UFCSET, UFENTR, COMAND, UF2MSG, SCPFCN	SCPHAS	SCENN, SCENC, UFICI, UFICR, UFICC, UFITN, UFITC
SCPRT	UF1PAR, UNQUOT, SCYORN, UF2MSG	SCREED	SCENN
SCPSEL	UFINIT, UFCSET, UFMSEL, UFMENU, COMAND, UF2MSG	SCENAR	SCENN, SCENC, UFITN, UFITC
SCREAL	UF1PAR, UF1FMT, UF1REL, UF2MSG	SCREED	None
SCREED	GETFIL, UF1PAR, SCRNI, SCHRD, SCCMP, SCYORN, SCPRT, SCREAL, SCMIS, UF2MSG	DATAIN	SCENN, UFITN
SCREPS	UF1PAR, UNQUOT, UF2MSG	SCREED	None
SCRID	UFINIT, UFCSET, UFENTR, COMAND	SCENAR	SCENC, UFICC, UFITN, UFITC
SCRNI	UF1PAR, UNQUOT, UF2MSG	SCREED	SCENC
SCSAVE	GETFIL, ARSORT, ARNEXT	SCENAR	ARCHN, ARCHC, SCENN, SCENC, UFITN
SCTEST	UF1DBL, UF1UPR	SCPARM	None
SCTIMS	UF1PAR, UF1FMT, UF1REL, UF2MSG	SCREED	None
SCTSEL	UFINIT, UFCSET, UFMFMT, UFENTR, COMAND, UF2MSG	SCENAR	SCENN, SCENC, UFICI, UFICR, UFICC, UFITN, UFITC

Table 2. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
SCYORN	UF1PAR, UNQUOT, UF2MSG	SCPRT, SCREED	None
UFCBAD	UF1IND	UFCEXE, UFCHLP	UFPCL, UFPCC
UFCDEF	UF1DBL	INIT	UFCCI, UFCCL, UFCCC
UFCEXE	UF1CEL, UFCHLP, UFCBAD	UFEDSP, UFMDSP, UFREPT	UFPCI
UFCFND	UF1DBL, UF1UPR, UF1FMT, UF1INT	UFCHLP, UFEDSP, UFMDSP, UFREPT	UFCCI, UFCCL, UFCCC
UFCHBL	UF1OPN	UFCHLP	UFPCI, UFPCC
UFCHLP	UFCHBL, UF1CEL, UF2PAG, UF1GET, UF1PAR, UFCFND, UFCBAD	COMAND, UFCEXE	UFPCI, UFPCL
UFCSET	None	DATAIN, ARCFCN, ARCHAN, ARCHTR, ARCMOD, ARFUNC, ARLRMU, ARPMOD, ARPOOL, ARPRSC, ARPUTL, ARRMOD, ARRSRC, ARSAVE, GETFIL, INIT, SCCSEL, SCPARM, SCPFCN, SCPHAS, SCPMOD, SCRID	UFCCI, UFCCL
UFECST	None	UFESET	UFICI, UFPCL, UFPCC
UFEDSP	UF1CEL, UFCEXE, UF2PAG, UF1GET, UF1PAR, UF1FMT, UF1INT, UFESET, UFCFND	UFENTR	UFICC, UFPCI, UFPCL, UFPCC
UFEIST	UF1FMT, UF1INT, UF1ICK	UFESET	UFICI, UFPCL, UFPCC
UFELST	UF1LST, UF1CEL, UF1IND, UF1NCL, UF1LCL, UF1CLR	UFENTR	UFICI, UFICR UFICC, UFPCI, UFPCC
UFENTR	UFELST, UFEDSP	ARCHAN, ARCMOD, ARFUNC, ARLRMU, ARPMOD, ARPOOL, ARPRSC, ARPUTL, ARRMOD, ARRSRC, GETFIL, SCPARM, SCPHAS, SCPMOD, SCRID	UFPCI

Table 2. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
UFERST	UF1FMT, UF1INT, UF1REL, UF1RCK	UFESET	UFICI, UFICR, UFPCL, UFPCC
UFESET	UFEIST, UFERST, UFECST	UFEDSP	UFICI, UFICR, UFICC, UFPCI, UFPCL, UFPCC
UFINIT	None	DATAIN, ARCFCN, ARCHAN, ARCHTR, ARCMOD, ARFUNC, ARLRMU, ARPMOD, ARPOOL, ARPRSC, ARPUTL, ARRMOD, ARRSRC, ARSAVE, GETFIL, INIT, SCCSEL, SCPARM, SCPFCN, SCPHAS, SCPMOD, SCRID	UFPCI, UFPCC
UFIT	UF1OPN, UF1WID	INIT	UFPCI, UFPCL, UFPCC
UFMDEF	UF1CEL	UFMENU	UFPCI, UFPCC
UFMDSP	UF1CEL, UFCEXE, UF2PAG, UF1GET, UF1PAR, UF1FMT, UF1INT, UFCFND	UFMENU	UFPCI, UFPCL, UFPCC
UFMENU	UFMLST, UFMDEF, UFMDSP, UFMRET	DATAIN, ARCFCN, ARCHTR, SCCSEL, SCPFCN	UFPCI
UFMFMT	None	ARCFCN, ARFUNC, ARLRMU, ARPUTL, INIT, SCPFCN	UFPCI
UFMLST	UF1LST, UF1CEL, UF1IND, UF1NCL, UF1LCL, UF1CLR	UFMENU	UFPCI, UFPCC
UFMRET	None	UFMENU	UFPCI, UFPCC
UFMSEL	None	ARCFCN, SCCSEL, SCPFCN	UFPCI
UFREPT	UF1CEL, UFCEXE, UF2PAG, UF1GET, UF1PAR, UFCFND	ARSAVE, INIT	UFPCI, UFPCL
UF1CEL	None	ARCFCN, ARPUTL, SCPFCN, UFCEXE, UFCHLP, UFEDSP, UFELST, UFMDEF, UFMDSP, UFMLST, UFREPT, UF1NCL	None

Table 2. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
UF1CLR	None	UFELST, UFMLST, UF2TXT	None
UF1DBL	None	GETFIL, SCTEST, UFCDEF, UFCFND	None
UF1ERS	None	UF2PAG	None
UF1FMT	None	ARCHI, ARPOL, ARRES, SCMIS, SCREAL, UFCFND, UFEDSP, UFEIST, UFERST, UFMDSP	None
UF1GET	UF1IND, UF1ROP	UFCHLP, UFEDSP, UFMDSP, UFREPT	None
UF1ICK	None	UFEIST	None
UF1IND	None	ARCFCN, ARPOOL, ARPRSC, ARPUTL, SCPFCN, UFCBAD, UFELST, UFMLST, UF1GET, UF1PUT	None
UF1INT	None	ARCHI, ARPOL, ARRES, SCMIS, UFCFND, UFEDSP, UFEIST, UFERST, UFMDSP	None
UF1LCL	None	UFELST, UFMLST	None
UF1LST	None	UFELST, UFMLST	None
UF1NCL	UF1CEL	UFELST, UFMLST	None
UF1OPN	None	GETFIL, UFCHBL, UFIT	None
UF1PAR	None	ARCHI, ARFCN, ARLRU, ARPOL, ARREAD, ARRES, SCCMP, SCHRD, SCMIS, SCPRT, SCREAL, SCREED, SCRNI, SCYORN, UFCHLP, UFEDSP, UFMDSP, UFREPT	None
UF1PUT	UF1IND	UF2PAG	None
UF1RCK	None	UFERST	None
UF1REL	None	ARPOL, SCMIS, SCREAL, UFERST	None

Table 2. (Concluded)

Routine	Routines Called	Calling Routines	Common Blocks
UF1ROP	None	UF1GET	None
UF1UPR	None	ARCONV, GETFIL, SCTEST, UFCFND	None
UF1WID	None	UFIT	None
UF2MSG	None	DATAIN, ARCFCN, ARCHAN, ARCHI, ARCMOD, ARFCN, ARFUNC, ARLRMU, ARLRU, ARPMOD, ARPOL, ARPOOL, ARPRSC, ARREAD, ARRES, ARRMOD, ARRSRC, GETFIL, SCCMP, SCHRD, SCMIS, SCPARM, SCPHAS, SCPMOD, SCPRT, SCREAL, SCREED, SCRNI, SCYORN	UFPCL, UFPCC
UF2PAG	UF1ERS, UF1PUT	UFCHLP, UFEDSP, UFMDSP, UFREPT	UFPCI, UFPCL, UFPCC
UF2TXT	UF1CLR	ARSAVE, INIT	UFPCI, UFPCC
UNQUOT	None	ARCHI, ARFCN, ARLRU, ARPOL, ARRES, SCCMP, SCHRD, SCMIS, SCPRT, SCRNI, SCYORN	None

2.5 DATAIN File Usage

Table 3 shows the logical device number assignments used by DATAIN for the reading and writing of files. DATAIN may access up to five file types during its execution. Some of the file activity is required by DATAIN, and some is optional and under control of the user. All DATAIN sessions require terminal input and display to read inputs at an interactive terminal. DATAIN reads and writes architecture and/or scenario files easily under the direction of the user. The specific architecture/scenario file to be accessed is named by the user during program execution. The format of these files is described in detail in Veatch (1986). DATAIN will not allow the user to destroy an existing architecture/scenario file by overwriting. (This activity can only be done outside of DATAIN by entering the appropriate system commands.) Finally, DATAIN reads two screen generation files. The Screen Generation Data File is

Table 3. DATAIN Logical Device Assignments

Logical Device Number	File	Type	Format
*	Terminal - Input and Display	Input/Output	A80
8	Architecture Files	Input/Output	A80
8	Scenario Files	Input/Output	A80
15	Screen-Generation Data File	Input	A80
15	Screen-Generation Help File	Input	A80

required. The Screen Generation Help File is not required for running DATAIN, but its absence would greatly reduce the 'user-friendly' aspects of DATAIN. It contains the Help, Explain, and Quit screens which are used by DATAIN to provide information on program usage and to define various terms. The two screen generation files are described in more detail below.

The Screen Generation Data File contains one record which is used to describe the physical characteristics of the terminal on which it is running and certain other aspects of the screens to be produced. Table 4 shows the fields that are expected to be on the first record of the Screen Generation Data File. The data are read by a FORTRAN 77, list-directed READ statement; therefore, the record is free-format.

The Screen Generation Help File contains all the auxiliary screens (Help, Explain, and Quit) that are used by DATAIN. These screens are placed in the file one after the other, and DATAIN locates an individual screen by its position within the file. Table 5 shows the various records that make up an individual screen. Each screen is delimited by a Screen Identification Record as the first record. All records in the file are 80-character card-images. Table 6 describes the format of the Screen Identification Record. This record is fixed-format with an asterisk ('*') in column one. Only Screen Identification Records have the asterisk in column one; the other records in the file have no restrictions. The requested screen is constructed by putting the Screen Header and Trailer Records at the top and bottom, respectively, of every page of the screen and filling each consecutive page of the screen with the next Screen Text Records until a new Screen Identification Record or the end of the file is encountered.

Table 4. Screen Generation Data File Record

Field	Description	Type
1	Number of Columns on Screen (80 Maximum)	INTEGER
2	Number of Lines on Screen	INTEGER
3	Null Field	INTEGER
4	Null Field	INTEGER
5	Length of List Delimiter (1 to 9)	INTEGER
6	List Delimiter String	CHARACTER*9
7	List Selection Symbol	CHARACTER*1

Table 5. Screen Generation Help File Organization

Record	Description
1	Screen Identification Record (defined by an asterisk '*' in column one)
Next m Records	Screen Header Record(s) where m is from Screen Identification Record
Next n Records	Screen Trailer Record(s) where n is from Screen Identification Record
Following Records	Screen Text Record(s) continue until next Screen Identification Record or the end of the file

Table 6. Screen Generation Help File Screen Identification Record

Columns	Description	Type
1	Asterisk ('*')	A1
2-10	Not Used	-
11-15	Number of Header Records in Screen	I5
16-20	Number of Trailer Records in Screen	I5
21-80	Not Used	-

3. MIREM PROGRAM

3.1 General Description of MIREM

The MIREM program implements the MIREM model, which was developed to evaluate the mission reliability of advanced electronics systems during the early development phases. These electronic systems contain fault-tolerant design aspects requiring new mathematical techniques to relate the overall mission reliability to the reliability of the electronic components.

The MIREM program consists of 82 FORTRAN 77 routines -a main routine (MIREM), a BLOCK DATA subprogram, 70 subroutines and 10 functions. This self-contained program implements all of the calculations described in Appendix A of Veatch (1986) and produces any selection of eight basic output reports:

1. Mission Completion Success Probability (MCSP) and Budget Report.
2. Phase-by-Phase MCSP Report.
3. Mean Time Between Critical Failures (MTBCF) Report.
4. Mean Time Between Function Failure (MTBFF) Report.
5. Line Replaceable Module/Line Replaceable Unit (LRM/LRU) Budget Report.
6. Repair Policy Report.
7. Testability factors - BIT option.
8. Testability factors - BIT MTBCF option.

The MIREM program also produces reports on the contents of the scenario and architecture input files. Examples of these reports are found in Appendix B. Finally, the MIREM program optionally creates a binary file that can be used for producing plots by the MPLOT program.

Section 3.2 of this chapter describes the basic program structure of the MIREM program. Section 3.3 briefly describes the function of each of the 82 routines. References to the MIREM equations in Appendix A of Veatch (1986) are supplied

when applicable. Section 3.4 displays the subprogram interdependencies and common block structure. Finally, Section 3.5 describes the program file usage.

3.2 MIREM Program Structure

This section describes the basic high-level program structure of the MIREM program. Section 3.2.1 deals with the main program; Section 3.2.2, with the reading of the input files; and Section 3.2.3, with the central computation performed by the model, the MCSP computation.

3.2.1 The MIREM Main Program

The main program MIREM controls the entire processing for the model. The MIREM tree diagram (Figure 10) illustrates the principal subprograms and their relationship to the main program. The MIREM flowchart (Figure 11) indicates the top-level processing performed by the main program.

The main program first reads the scenario file and architecture file. See Section 4.3 of Veatch (1986) for a description of the formats of these files. An example of these files is found in Appendix A. The routines SCREAD and HWREAD read the scenario file and architecture file, respectively. The routine CROSS then checks the correctness of the architecture file. If errors are found, processing stops with indicative error messages.

Nine routines perform preprocessing computations for various options. The GETF routines determine the basic function set used in the computations, and GETREQ determines basic requirements for each pool. The repair option requires four routines to be run: SERIES, REPCHK, RSORT, and RTABLE. SERIES determines if all chains are series chains. REPCHK checks the consistency of the pool requirements, the number of branches, and the minimum level of repair. RSORT sorts the resources table to prepare a lookup table of probabilities for MTTR computations. The testability options require the computation of the probability of undetected failures occurring on a required resource, performed by the GETRHO routine. PSTART opens the plot file if any plot outputs are desired. Finally, NCFULL precalculates the set of requirements for the full MCSP algorithm.

The main program may produce any of eight reports, depending on options found on the COMPUTE card in the scenario file. Table 7 indicates the report produced with each computation option, together with the top-level routine responsible for producing the report. As suggested in Figure 11, two reports - Phase-by-Phase MCSP and MTBCF - are independent of the total

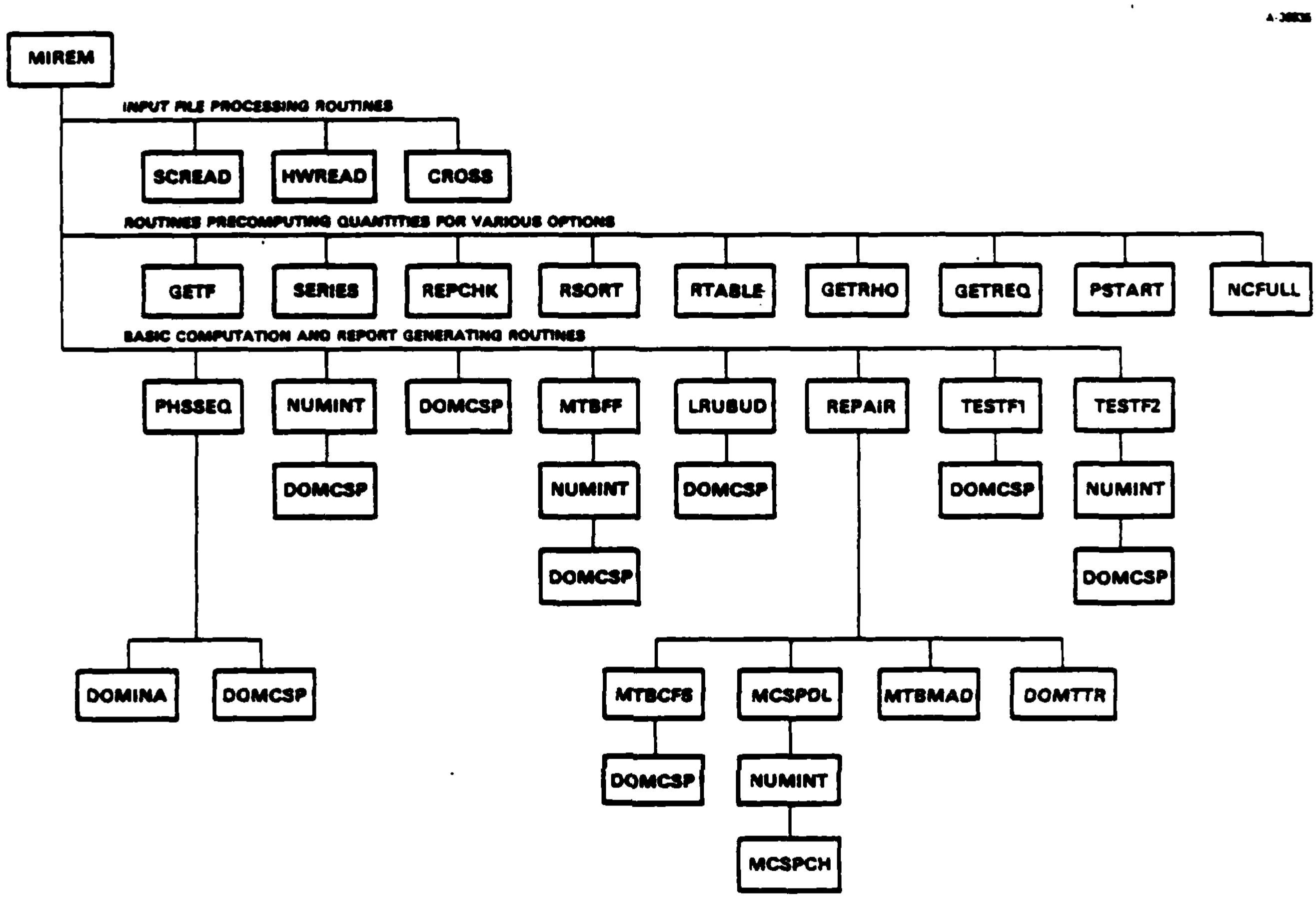


Figure 10. MIREM Tree Diagram - Main Program.

operating time. In addition, the Repair Policy Report is presently run only for the first total operating time used. Otherwise, the other five options may be run with a variety of total operating times in one run in order to measure the impact of total operating time on the reliability measures computed.

The MCSP and Budget Report is produced by the routine DOMCSP. The routine DOMCSP performs the basic MCSP calculation. The DOMCSP routine, which supports all the computation, will be described in detail in Section 3.2.3. The Phase-by-Phase MCSP Report is produced by the routine PHSSEQ, which requires phase-by-phase calls to the routine DOMINA to test for phase domination, and to DOMCSP to obtain phase-specific MCSP calculations. The MTBCF Report is performed by the NUMINT routine, which numerically integrates the system life distribution (MCSP) using successive calls to DOMCSP. The MTBFF routine prepares an MTBFF Report, which calls NUMINT once for each function to obtain function-specific MTBCF computations. The LRM/LRU Budget

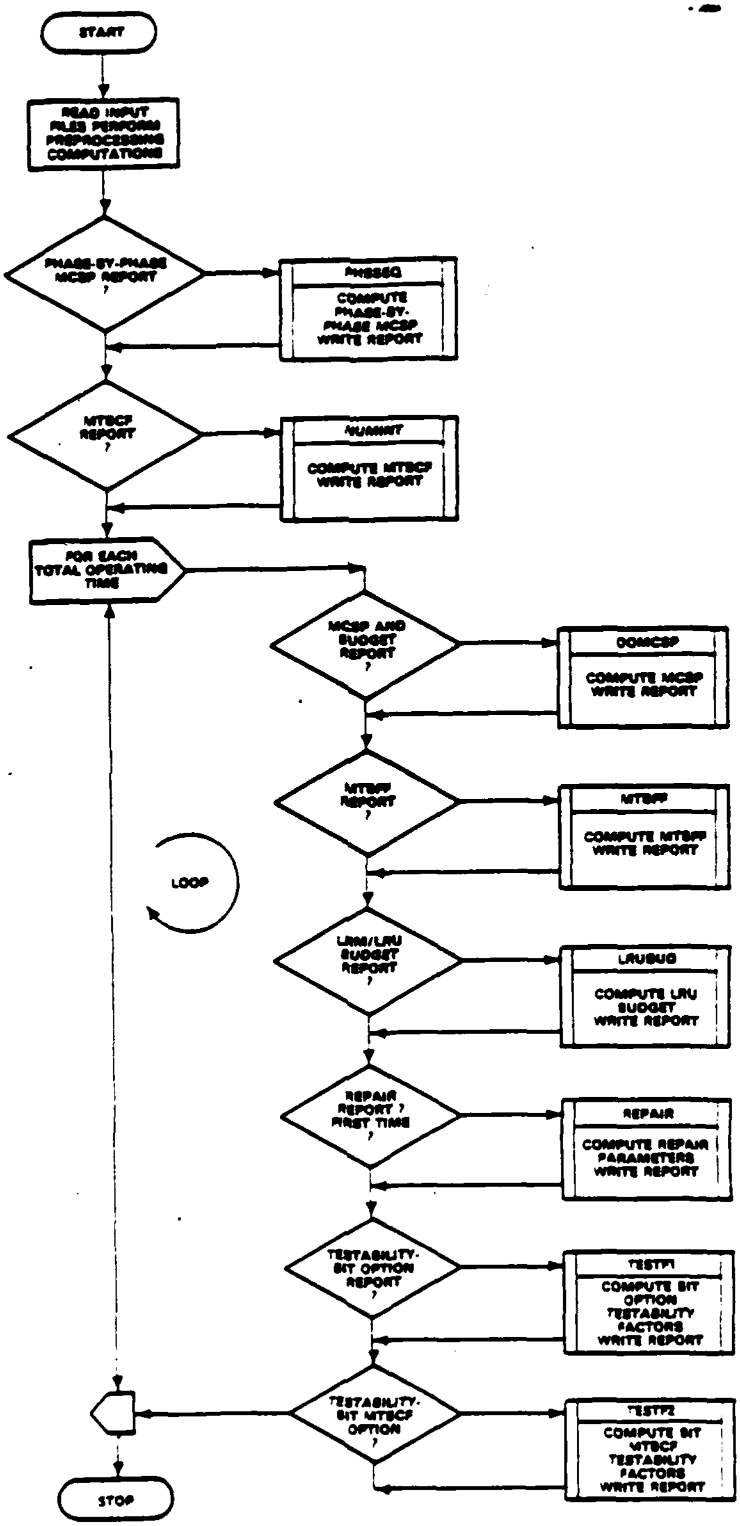


Figure 11. MIREM Flowchart - Main Program.

Table 7. MIREM Computation Options

Compute Option in Scenario File	Routine Responsible	Report Produced
MCSP	DOMCSP	MCSP and Budget Report - Breakdown of MCSP by each chain pair and by each pool or pool type contribution to the MCSP
PHASE-BY-PHASE	PHSSEQ	Phase-by-Phase MCSP Report - Compute upper and lower bounds to MCSP for each phase of a multiphase mission
MTBCF	NUMINT	MTBCF Report - Compute Mean Time Between Critical Failure and system failure resiliency
MTBFF	MTBFF	MTBFF Report - Determine Mean Time Between Function Failure for each function performed by the system
LRU	LRUBUD	LRM/LRU Budget Report - Determine contribution of each LRM/LRU to the MCSP
REPAIR	REPAIR	Repair Policy Report - Compute effect of various repair policies on reliability measures
BIT	TESTF1	Testability Factors Report - Bit Options
FULLBIT	TESTF2	Testability Factors Report - Full Bit Option

Report is performed by LRUBUD, which finds the contribution of each LRM/LRU to the system reliability with LRU-by-LRU calls to DOMCSP. The Repair Policy Report is produced by the REPAIR routine, which calls the MTBCFS, MCSPDL, MTBMAD, and DOMTTR routines for some of the more complicated repair policy computations. These routines in turn call the DOMCSP, NUMINT, and MCSPCH routines. The BIT Option Testability Factors Report is produced by the TESTF1 routine, which calls DOMCSP once for each pool. Finally, the Full BIT Option Testability Factors Report is produced by the TESTF2 routine, which is built on the NUMINT routine. The utility routines IMMREP and MTBFCA compute the immediate repair MTBCF and the system MTBF, respectively, for the various reports. All the computation options except the two Testability Factor routines can produce data records on the plot file for use by the plot program.

3.2.2 Reading the Input Files

The SCREAD and HWREAD routines read the input files. Examples of the scenario and architecture files are found in Appendix A.

The subroutine SCREAD reads the scenario file. The routines called by SCREAD are indicated by the SCREAD tree diagram (Figure 12). The SCDFLT routine provides default values for the scenario file parameters. The HARDWARE, REPSEQUENCE, TIME, RUNID, PRINT, and PHASE cards are read by the routines SCHARD, SCREPR, SCTIME, SCRUN, SCPRNT, and SCMSSN, respectively. The COMPUTE and PLOT cards are both read by the SCCOMP routine. The SIMULTANEOUS and QUICK cards are both interpreted by the routine SCQUES; and the SCALE and TMAINTENANCE cards, by the routine SCPNUM. The routine SCPRNT also calls SCQUES as part of the interpretation of PRINT cards. Finally, the routine SCDUMP writes out a scenario file report. These routines make heavy use of the utility routines GETTOK, ERMSG1, IDECOD, and RDECOD.

The subroutine HWREAD reads in the architecture file, utilizing the routines indicated in the HWREAD tree diagram (Figure 13). FUNCTION, LRU, RESOURCE, CHAIN, GROUP, and POOL cards are read and interpreted by the routines HWFUNC, HWLRU, HWRESO, HWCHAI, HWGROU, and HWPOOL, respectively. HWFAIL computes the pool failure rate given a list of resources comprising the pool. The routine HWREPO produces the pool report of the architecture file report, and HWDUMP produces the remainder of the architecture file report. HWUTIL reads the function utilizations for pools or groups. These routines also use the same utilities that are heavily used by the SCREAD subroutine.

Note that the scenario file must be read before the architecture file for two reasons: (a) the scenario file may contain the name of the architecture file to read on a HARDWARE card, and (b) the scenario file contains the scale parameter for uniform multiplication of all pool failure rates. The values of

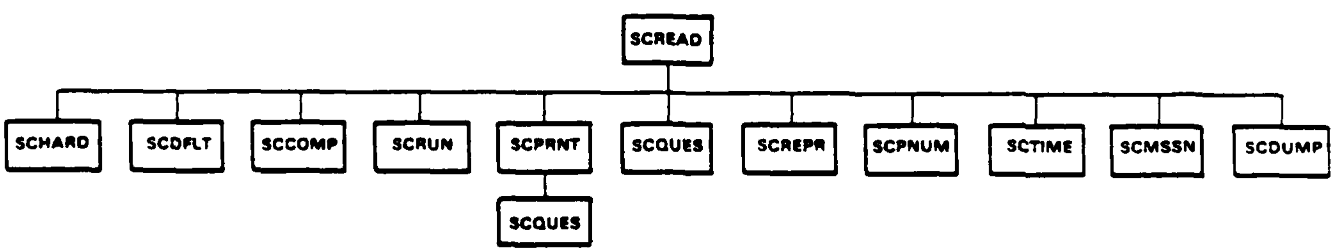


Figure 12. SCREAD Tree Diagram - Read Scenario File.

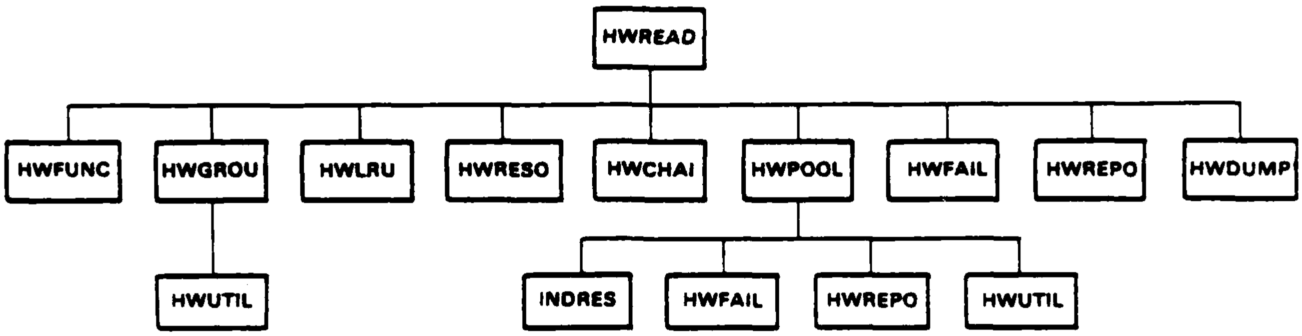


Figure 13. HWREAD Tree Diagram - Read Architecture File.

the pool failure rates appearing on the architecture file report differ from the pool failure rates read from the architecture file by a factor given in the SCALE card in the scenario file. This factor allows a quick rescaling of the relative pool failure rates read by the MIREM model.

3.2.3 MCSP Computation

The MCSP computation, required by all reports, will be described in detail. All equations cited are from Veatch (1986). The routine DOMCSP organizes the computation of the MCSP. Figures 14 and 15, respectively, display the DOMCSP tree diagram and the DOMCSP flowchart. This routine organizes the computation of each chain pair MCSP; their product is the system MCSP (see Equation 17). For series chains without groups, MCSP is calculated by the routine CHAIN (Equation 6). It calls the routine POOL once for each pool in the chain. Pool probabilities, Equations 4 or 5, are precomputed by the routine POOLTB and stored for quick lookup by the routine POOL. The routine POOLX computes pool probabilities not found on the table. POOLX and POOLTB use the POOLRS and POOLAC functions to compute probabilities for standby and active pools, respectively.

For series chains with groups (cascading chains), MCSP is calculated by the GROUP subroutine (Equation 7). The routine POOL is called once for each group or pool in the chain.

MCSP calculations for parallel chains are more complicated (Equation 8). The CHAIN routine also can be used to obtain the factors $\Pr\{UP^k(F)\}$, and $\Pr\{UP^k(S,N,C)\}$ for k = 1 or 2. The $\Pr\{UP^{1+2}(S) | UP^1(F), UP^2(F)\}$ factor is computed by the routine PUP12S using Equation 16 via calls to the POOL routine. The most complicated term is $\Pr\{UP^{1+2}(N,C)\}$. The CFFUNC routine is

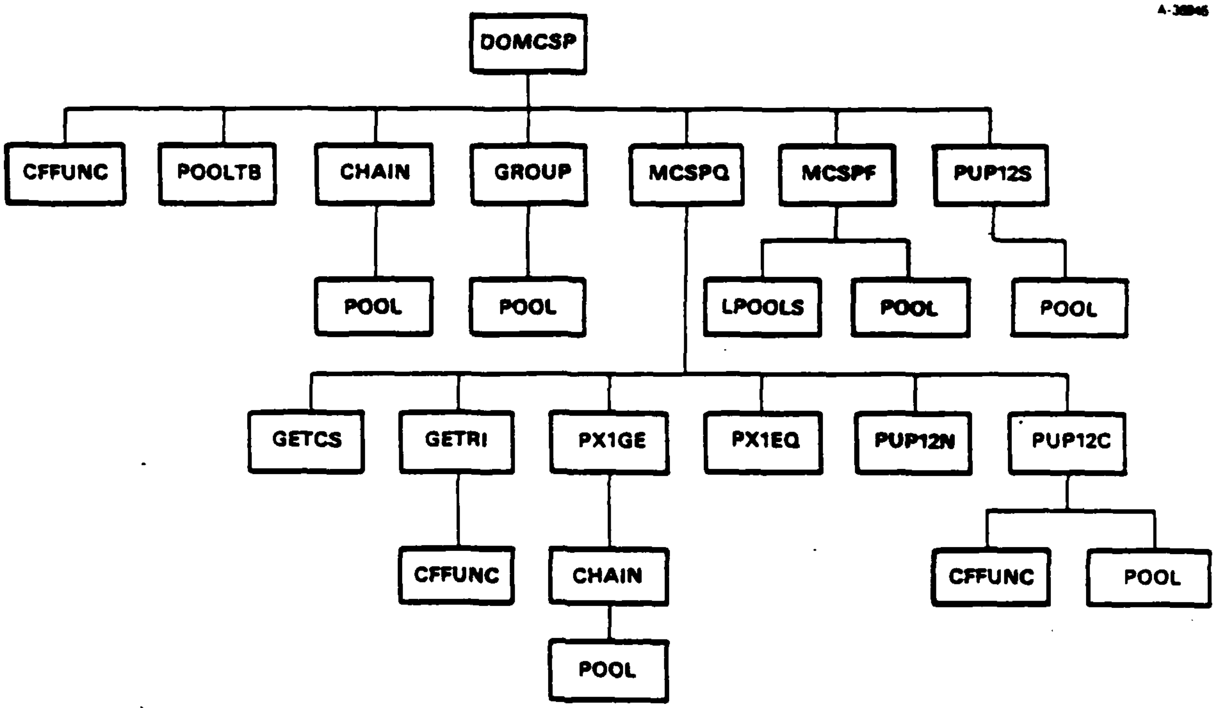


Figure 14. DOMCSP Tree Diagram - Compute MCSP.

used to first partition the set of functions into distinct subsets determined by the use of each function by the chain pair. Two algorithms are available, controlled by the QUICK parameter in the scenario file. The "full" algorithm, using MCSPF and Equation 14, was developed by Dr. Robert Foley and is new to this release of MIREM. It uses precomputed requirements vectors from the NCFULL routine. The routine LPOOLS is invoked to obtain lists of pool indices to process in the POOL routine.

The MCSPQ routine calculates the term $\Pr\{UP^{1+2}(N,C)\}$ using the "quick" algorithm (Equation 9), with processing as indicated in the MCSPQ flowchart (Figure 16). First, the GETCS routine establishes a list of the type C pools, and the GETRI routine calculates and stores the r_i and $r_{max,i}$ terms (Equations 10c and 10d). The PX1GE routine computes and stores the distribution of the $\underline{X}^k$ vectors in Equation 9, k = 1 or 2, where the $\underline{X}^k$ indicates which functions can be supported on the type N pools on chain k in a chain pair. PX1GE calls the routine CHAIN with a different function set for each value of $\underline{X}^k$. The routine PX1EQ converts the $\underline{X}^h$ distribution into marginal density values ($\Pr\{\underline{X}^k = \underline{x}\}$) using the law of total probability.

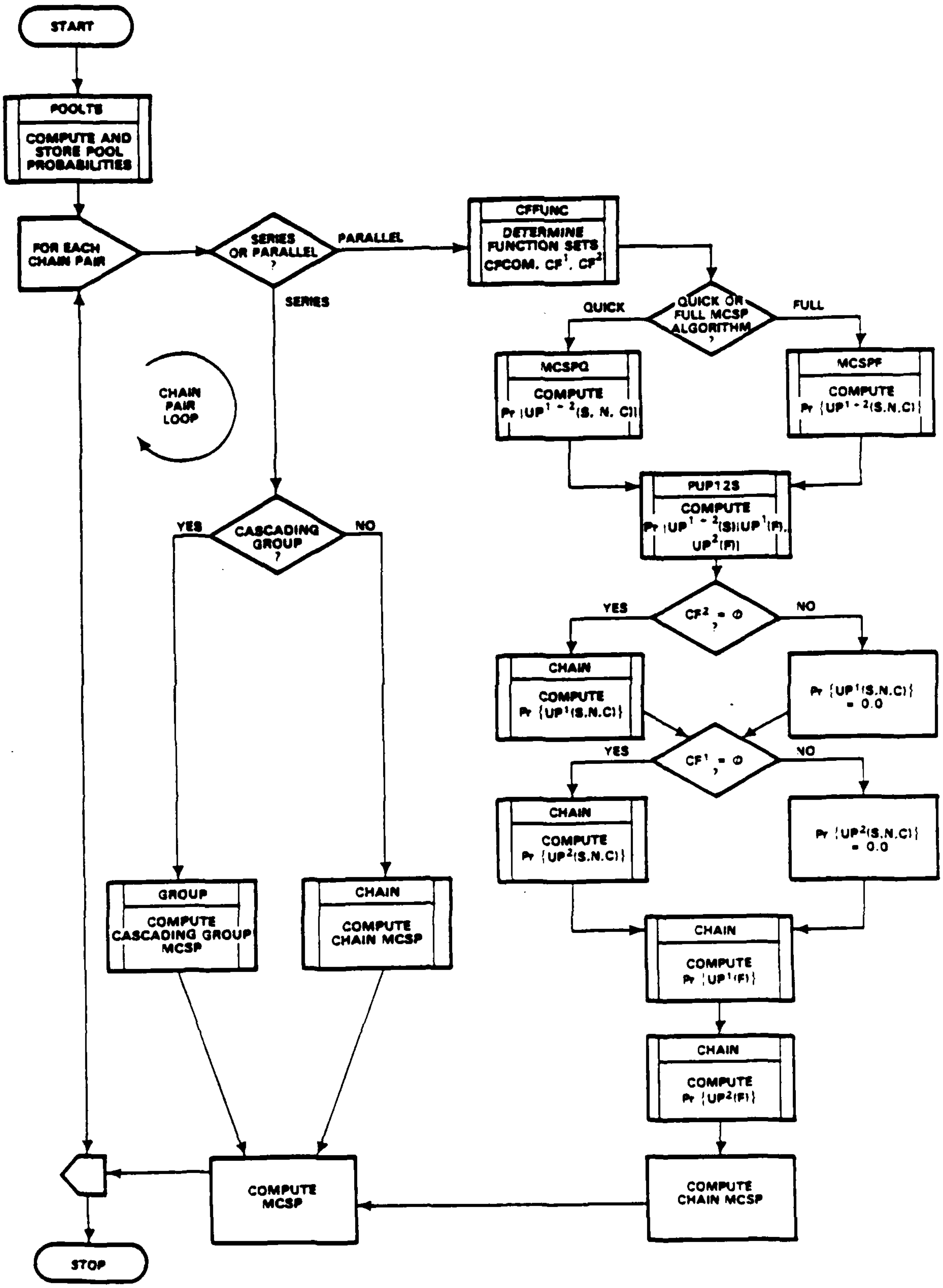


Figure 15. DOMCSP Flowchart - Compute MCSP.

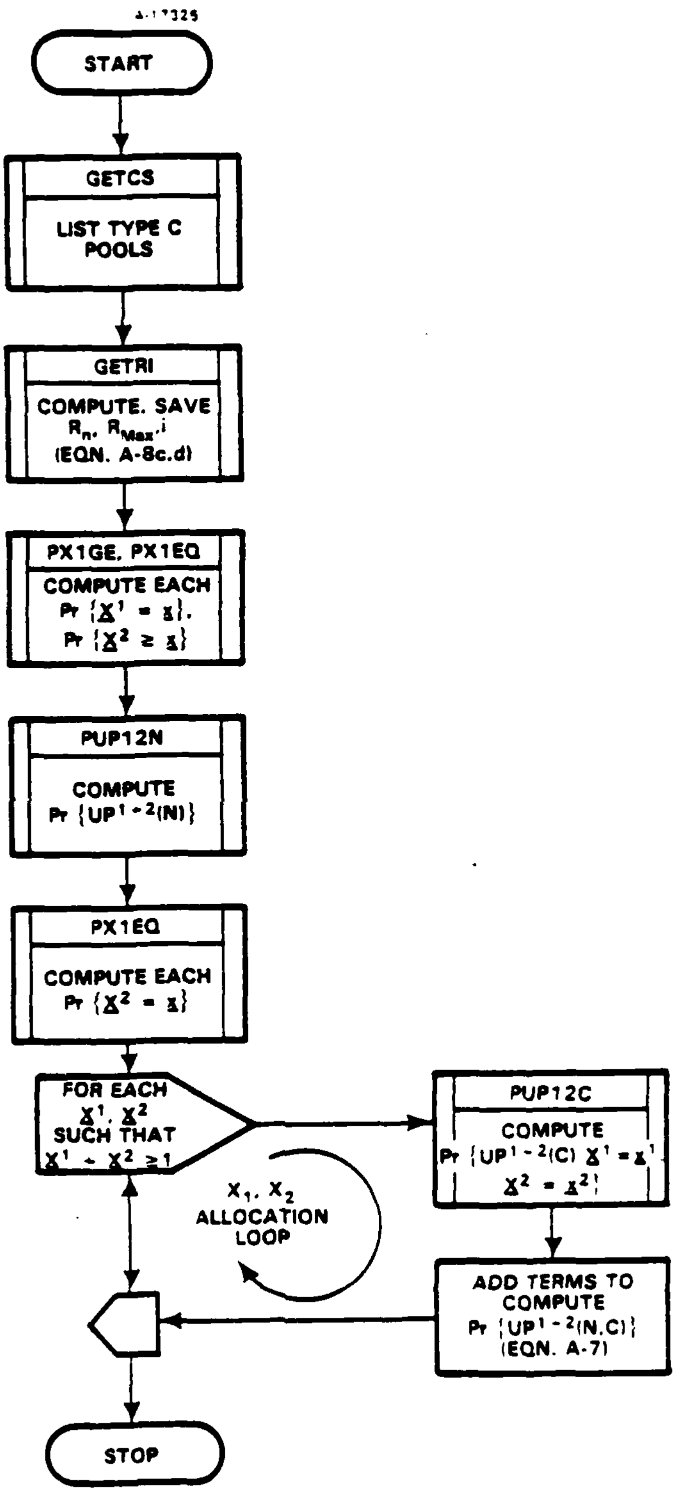


Figure 16. MCSPQ Flowchart - Compute Pr{UP^{1+2}(N,C)}.

The routine PUP12N uses the outputs from PX1GE and PX1EQ to compute $Pr\{UP^{1+2}(N)\}$ as follows:

$$Pr\{UP^{1+2}(N)\} = \sum_{\underline{x}=\underline{0}}^{\underline{1}} Pr\{\underline{X}^1 = \underline{x}\}\ Pr\{\underline{X}^2 \geq \underline{1} - \underline{x}\} \qquad (1)$$

Finally, the PUP12C routine uses Equation 13 to compute $Pr\{UP^{1+2}(C) | \underline{X}^1 = \underline{x}^1, \underline{X}^2 = \underline{x}^2\}$ for all values of $\underline{x}^1$ and $\underline{x}^2$.

3.3 MIREM Subroutine Descriptions

The 51 subprograms comprising the MIREM program are briefly described, in alphabetical order, in Table 8. Where applicable, references to Appendix A are supplied.

Table 8. MIREM Routine Descriptions

Routine	Description
MIREM	Main Program. Read in and check the architecture file and the scenario file. Perform computations. Produce any selection of eight basic output reports: (1) MCSP and BUDGET Report, (2) PHASE-BY-PHASE MCSP Report, (3) MTBCF Report, (4) MTBFF Report, (5) LRM/LRU Budget Report, (6) Repair Policy Report, (7) Testability Factors - BIT option, or (8) Testability Factors - Full BIT Option.
BLOCK DATA	Initialize variables in COMMON blocks to their default values.
CEIL	Obtain smallest integer greater than or equal to a given real number.
CFFUNC	Partition a given function set into distinct subsets indicating how a given parallel chain pair uses the functions. (Appendix A.4 of Veatch, 1986, and discussion of CFCOM, CF^k, $FCOM_\ell$, and F_ℓ^k).
CHAIN	Determine probability that a given chain can support the functions on the specified pool types. (Equation 6)
CHAIN2	Determine chain probability for cases when parallel chains are treated as series chains.
CROSS	Check pool data for consistency across pool pairs in parallel chains.

Table 8. (Continued)

Routine	Description
DMTTR	Look up MTTR from table prepared by RTABLE routine.
DOMCSP	Compute MCSP for a multiphase mission.
DOMINA	Determine whether one mission phase dominates another mission phase. (See discussion of dominated in Appendix A.5 of Veatch, 1986)
DOMTTR	Compute MTTR for a repair policy, given a multiple repair policy.
ERMSG1	Print first line of error messages written when reading input files.
FCNLST	Prepare formatted function list for printout.
FCNON	Format a string of function indices selected from list for printout purposes.
FCNPRT	Print function list.
FCNSTR	Write string of function indices for printout purposes.
GETCS	List indices of type 'C' pools that are required by a particular chain number.
GETF	Determine set of functions utilized in some phase.
GETREQ	Determine pool requirements, ignoring parallel chain structure.
GETRHO	Compute probability an undetected failure occurs on a required resource.
GETRI	Compute requirements for type 'C' pools within a particular chain. (Equations 10c and 10d)
GETTOK	Return next set of characters to process while reading in the MIREM input files.
GROUP	Compute probability a chain can support the functions for a cascading group series chain.
HEADER	Print top line of page for reports written by MIREM.
HWCHAI	Read and interpret CHAIN cards in architecture file.
HWDUMP	Print architecture file report, excluding pool report.

Table 8. (Continued)

Routine	Description
HWFAIL	Determine pool failure rates from list of resources comprising the pool.
HWFUNC	Read and interpret FUNCTION cards in architecture file.
HWGROU	Read and interpret GROUP cards in architecture file.
HWLRU	Read and interpret LRU cards in architecture file.
HWPOOL	Read and interpret POOL cards in architecture file.
HWREAD	Read the architecture file.
HWREPO	Print pool cross-reference report, one pool at a call.
HWRESO	Read and interpret RESOURCE cards in architecture file.
HWUTIL	Read and interpret function utilization by pool/group values in architecture file.
IDECOD	Decode a character string into an integer number.
IMMREP	Compute immediate repair MTBCF.
INDRES	Find resource index corresponding to given resource number.
LPOOLS	Obtain list of pool indices for a parallel chain for Full MCSP algorithm.
LRUBUD	Prepare an LRM/LRU budget report. (Appendix A.9 of Veatch, 1986)
MCSPCH	Compute MCSP, treating each parallel chain as two series chains.
MCSPDL	Compute MCSP for degraded level repair policy.
MCSPF	Use Dr. Foley's full algorithm to compute the probability that critical functions are up on type 'N' and 'C' pools in a parallel chain. (Equation 14)
MCSPQ	Use quick algorithm to compute the probability that critical functions are up on type 'N' and 'C' pools in a parallel chain. (Equation 9)
MTBCFS	Compute MTBCF for scheduled maintenance repair policy.

Table 8. (Continued)

Routine	Description
MTBFCA	Compute Mean Time Between Failures in terms of individual pool failure rates: $$MTBF = \frac{1}{\sum_{pool,i} \lambda_i \, c_{max,i}}$$ λ_i = failure rate on each branch of pool i $c_{max,i}$ = number of branches (maximum capacity) for pool i
MTBFF	Prepare a MTBFF (Mean Time Between Function Failure) report.
MTBMAD	Compute MTBMA for degraded-level repair policy.
NCFULL	Precompute requirements for pools in parallel chains. Used by Full MCSP algorithm.
NUMINT	Compute MTBCF under a deferred repair policy by numerical integration of the system life distribution. (Implements Appendix A.6 of Veatch, 1986)
PHSSEQ	Compute upper and lower bounds on the probability that critical functions are up after each phase of a mission. (Implements Appendix A.5 of Veatch, 1986)
POOL	Return the probability that the capacity of a pool at a given time is greater than a given requirement. (Equation 4 or 5)
POOLAC	Compute pool probability for an active pool.
POOLRS	Compute pool probability for a standby pool.
POOLTB	Compute the pool probabilities for each requirement at a given time and store the results in a table for quick look-up by the POOL routine.
POOLX	Compute a pool probability for a requirement not placed in the table.
PSTART	Open plot file. Place descriptor information in first record.
PUP12C	Compute the conditional probability that either chain can support the functions on type 'C' pools, given the function status on type 'N' pools.

Table 8. (Continued)

Routine	Description
PUP12N	Compute the probability that either chain in a chain pair will support the functions on the type 'N' pools. ($Pr\{UP^{1+2}(N)\}$ factor in Equation 8)
PUP12S	Compute the conditional probability that critical functions are up on type 'S' pools, given that they are up on both the primary and secondary chain for type 'F' pools. (Equation 16)
PX1EQ	Compute the marginal distribution of $\underline{X}^k$, the set of functions that are up with respect to the type N pools on a chain. (See discussion below Equation 9)
PX1GE	Compute the cumulative distribution of the set of functions that are up with respect to the type N pools on a chain. (See discussion below Equation 9)
RDECOD	Decode a character string into a floating point number.
REPAIR	Prepare repair policy report. (Appendix A.8 of Veatch, 1986)
REPCHK	Check consistency of pool requirements, pool repair levels and pool number of branches. Create dummy function for MTTR computations.
RSORT	Sort resource list by increasing MTTR (bubble sort).
RTABLE	Create look-up table for determining MTTR values.
SCCOMP	Read and interpret COMPUTE cards in the scenario file.
SCDFLT	Provide default values for parameters of scenario file.
SCDUMP	Print the scenario file report.
SCHARD	Read and interpret HARDWARE cards in the scenario file.
SCMSSN	Read and interpret PHASE cards in the scenario file.
SCPNUM	Read a positive floating point number from a card in the scenario file. (Used with TMAINTENANCE and SCALE cards of the scenario file)
SCPRNT	Read and interpret PRINT cards from the scenario file.
SCQUES	Read the keyword 'YES' or 'NO' on cards in the scenario file. (Used to read SIMULTANEOUS and QUICK cards in the scenario file)

Table 8. (Concluded)

Routine	Description
SCREAD	Read the scenario file.
SCREPR	Read and interpret REPSEQUENCE card in scenario file.
SCRUN	Read and interpret RUNID cards in the scenario file.
SCTIME	Read and interpret TIME cards in the scenario file.
SERIES	Determine if each chain is a series chain for repair policy report.
TESTF1	Prepare Testability Factors Report - BIT Option. (See Appendix A.7 of Veatch, 1986)
TESTF2	Prepare Testability Factors Report - Full BIT Option. (See Appendix A.7 of Veatch, 1986)

3.4 MIREM Subprogram Interdependencies and Common Block Usage

This section identifies the interdependencies between subprograms in terms of transfer of control and common storage. Table 9 lists for each program the routines it calls, the routines calling it, and the common blocks used. Common block parameter descriptions may be found in the source code for the BLOCK DATA routine.

3.5 MIREM File Usage

As shown in Table 10, the MIREM program uses four logical device numbers for files - the scenario file, the architecture file, the MIREM printed reports, and the plot file. The operating system must assign file names to the logical units associated with the scenario file and the MIREM output files. If the MIREM output files are not directed to specific file names, system defaults will be used. The architecture file may also be assigned by the operating system. Another option is for the scenario file to contain a HARDWARE card with the name of the architecture file. In this case, the MIREM program itself will assign the architecture file to logical device 5 for the operating system.

The two input files are expected in card image records (80-character records). The output record consists of up to 133-character records, including a carriage control.

Table 9. MIREM Subprogram Interdependencies and Common Block Usage

Routine	Routines Called	Calling Routines	Common Blocks
MIREM	CROSS, DOMCSP, GETF, GETREQ, GETRHO, HWREAD, LRUBUD, MTBFF, NCFULL, NUMINT, PHSSEQ, PSTART, REPAIR, REPCHK, RSORT, RTABLE, SCREAD, SERIES, TESTF1, TESTF2	None (Main Program)	None
BLOCK DATA	None	None	GROUPS, HW, HWNAME, HWPT, HWREPA, HWRSRC, MISSIO, REQFUL, RUNCH, TBLEPO, TESTPA
CEIL	None	CHAIN, PUP12C, GETREQ, PUP12S, CHAIN2, GROUP, GETRHO, MTBCFS, NCFULL, REPCHK	None
CFFUNC	None	DOMCSP, GETRI, PUP12C, GETRHO, NCFULL	HW
CHAIN	CEIL, POOL	DOMCSP, PX1GE	HW, HWPT, MISSIO
CHAIN2	CEIL, POOL	MCSPCH	HW, HWPT
CROSS	None	MIREM	HW, HWPT
DMTTR	None	DOMTTR	HWREPA, HWRSRC, REMTTR
DOMCSP	CFFUNC, CHAIN, FCNPRT, GROUP, HEADER, IMMREP, MCSPF, MCSPQ, POOLTB, PUP12S	MIREM, LRUBUD, MTBFF, NUMINT, PHSSEQ, TESTF1	GROUPS, HW, HWNAME, MISSIO
DOMINA	None	PHSSEQ	None
DOMMTR	DMTTR	REPAIR	None

Table 9. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
ERMSG1	None	HWCHAI, HWFUNC, HWGROU, HWLRU, HWPOOL, HWREAD, HWRESO, HWUTIL, SCCOMP, SCHARD, SCMSSN, SCPNUM, SCPRNT, SCQUES, SCREAD, SCRUN, SCTIME	None
FCNLST	None	FCNON, FCNPRT, MCSPF	None
FCNON	FCNLST, FCNSTR	HWDUMP, MCSPQ, SCDUMP	None
FCNPRT	FCNLST, FCNSTR	DOMCSP	None
FCNSTR	None	MCSPF, FCNON, FCNPRT	None
GETCS	None	MCSPQ	HW, HWPT
GETF	None	MIREM	HW, MISSIO
GETREQ	CEIL	MIREM	HW, HWPT, HWREPA
GETRHO	CEIL, CFFUNC	MIREM	GROUPS, HW, HWPT, HWREPA, TESTPA
GETRI	CFFUNC	MCSPQ	HW, MISSIO
GETTOK	None	HWCHAI, HWFUNC, HWGROU, HWLRU, HWPOOL, HWREAD, HWRESO, HWUTIL, SCCOMP, SCHARD, SCMSSN, SCPNUM, SCPRNT, SCQUES, SCREAD, SCRUN, SCTIME	None
GROUP	CEIL, POOL	DOMCSP	GROUPS, HW, HWPT, MISSIO
HEADER	None	DOMCSP, HWDUMP, HWREPO, LRUBUD, MTBCFS, MTBFF, NUMINT, PHSSEQ, SCDUMP, REPAIR, TESTF1, TESTF2	RUNCH
HWCHAI	ERMSG1, GETTOK, IDECOD	HWREAD	HW, HWNAME, HWRSRC

Table 9. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
HWDUMP	FCNON, HEADER	HWREAD	GROUPS, HW, HWNAME, HWPT, HWREPA, HWRSRC
HWFAIL	None	HWPOOL, HWREAD	HW, HWREPA, HWRSRC
HWFUNC	ERMSG1, GETTOK	HWREAD	HW, HWNAME
HWGROU	ERMSG1, GETTOK, HWUTIL, IDECOD	HWREAD	GROUPS, HW, HWNAME, HWPT
HWLRU	ERMSG1, GETTOK	HWREAD	HW, NWNAME
HWPOOL	ERMSG1, GETTOK, HWFAIL, HWREPO, HWUTIL, IDECOD, INDRES, RDECOD	HWREAD	HW, HWNAME, HWPCT, HWREPA, TESTPA
HWREAD	ERMSG1, GETTOK, HWCHAI, HWDUMP, HWFAIL, HWFUNC, HWGROU, HWLRU, HWPOOL, HWREPO, HWRESO	MIREM	HW
HWREPO	HEADER	HWPOOL, HWREAD	HW, HWNAME, HWPT, HWREPA, HWRSRC, TESTPA
HWRESO	ERMSG1, GETTOK, IDECOD, RDECOD	HWREAD	HWNAME, HWREPA, HWRSRC
HWUTIL	ERMSG1, GETTOK, RDECOD	HWGROU, HWPOOL	HW
IDECOD	None	HWCHAI, HWGROU, HWPOOL, HWRESO, SCMSSN	None
IMMREP	None	DOMCSP, MTBFF, PHSSEQ, REPAIR	None
INDRES	None	HWPOOL	HWRSRC
LPOOLS	None	MCSPF, NCFULL	HW, HWPT
LRUBUD	DOMCSP, HEADER	MIREM	HW, HWNAME, MISSIO

Table 9. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
MCSPCH	CHAIN2	NUMINT	HW
MCSPDL	NUMINT, POOLX	REPAIR	HW, HWPT, HWREPA
MCSPF	FCNLST, FCNSTR, LPOOLS, POOL	DOMCSP	HW, REQFUL
MCSPQ	FCNON, GETCS, GETRI, PUP12C, PUP12N, PX1EQ, PX1GE	DOMCSP	HW
MTBCFS	CEIL, DOMCSP	REPAIR	None
MTBFCA	None	MTBFF, NUMINT, REPAIR, TESTF2	HW
MTBFF	DOMCSP, HEADER, IMMREP, MTBFCA, NUMINT	MIREM	HW, HWNAME
MTBMAD	NUMINT	REPAIR	HW
NCFULL	CEIL, CFFUNC, LPOOLS	MIREM, PHSSEQ	HW, HWPT, REQFUL
NUMINT	DOMCSP, HEADER, MCSPCH, MTBFCA, POOLTB	MIREM, MTBFF, MCSPDL, MTBMAD, TESTF2	HW
PHSSEQ	DOMCSP, DOMINA, HEADER, IMMREP, NCFULL	MIREM	HW, HWNAME, MISSIO
POOL	POOLX	CHAIN, PUP12C, MCSPF, PUP12S, CHAIN2, GROUP	TBLEPO
POOLAC	None	POOLX	None
POOLRS	None	POOLTB, POOLX	None
POOLTB	POOLRS	DOMCSP, NUMINT	HW, TBLEPO, HWPT
POOLX	POOLAC, POOLRS	POOL, MCSPDL	HWPT
PSTART	None	MIREM	RUNCH
PUP12C	CEIL, CFFUNC, POOL	MCSPQ	HW, MISSIO

Table 9. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
PUP12N	None	MCSPQ	None
PUP12S	CEIL, POOL	DOMCSP	HW, HWPT, MISSIO
PX1EQ	None	MCSPQ	None
PX1GE	CHAIN	MCSPQ	HW
RDECOD	None	HWPOOL, HWRESO, HWUTIL, SCMSSN, SCPNUM, SCTIME	None
REPAIR	DOMMTR, HEADER, IMMREP, MCSPDL, MTBCFS, MTBFCA, MTBMAD	MIREM	None
REPCHK	CEIL	MIREM	HW, HWPT, HWREPA
RSORT	None	MIREM	HWREPA, HWRSRC
RTABLE	None	MIREM	HWREPA, HWRSRC, REMTTR
SCCOMP	ERMSG1, GETTOK	SCREAD	None
SCDFLT	None	SCREAD	None
SCDUMP	FCNON, HEADER	SCREAD	HWNAME, MISSIO
SCHARD	ERMSG1, GETTOK	SCREAD	None
SCMSSN	ERMSG1, GETTOK, IDECOD, RDECOD	SCREAD	HWNAME, MISSIO
SCPNUM	ERMSG1, GETTOK, RDECOD	SCREAD	None
SCPRNT	ERMSG1, GETTOK, SCQUES	SCREAD	None
SCQUES	ERMSG1, GETTOK	SCREAD, SCPRNT	None
SCREAD	ERMSG1, GETTOK, SCCOMP, SCDFLT, SCDUMP, SCHARD, SCMSSN, SCPNUM, SCPRNT, SCQUES, SCREPR, SCRUN, SCTIME	MIREM	None

Table 9. (Concluded)

Routine	Routines Called	Calling Routines	Common Blocks
SCREPR	ERMSG1, GETTOK	SCREAD	None
SCRUN	ERMSG1, GETTOK	SCREAD	RUNCH
SCTIME	ERMSG1, GETTOK, RDECOD	SCREAD	None
SERIES	None	MIREM	GROUPS, HW, HWPT, HWREPA
TESTF1	DOMCSP, HEADER	MIREM	HW, TESTPA
TESTF2	HEADER, MTBFCA, NUMINT	MIREM	HW, TESTPA

Table 10. Logical Device Assignments

Logical Device Number	File	Type	Format
4	Scenario File	Input	A80
5	Architecture File	Input	A80
6	MIREM Printed Reports	Output	A133
20	Plot File	Output	Binary

4. MPLOT PROGRAM

4.1 General Description of MPLOT

The MPLOT program allows a user to plot selected MIREM data. MIREM generates a file that contains plot data selected through the scenario file plot card. A user also may control where the plots are displayed; i.e., a terminal or a hard-copy device.

The MPLOT program consists of 51 FORTRAN 77 routines: a main program (MPLOT), 47 subroutines and three functions. MPLOT supplies graphics output through the DI-3000™ graphics package.

Section 4.2 describes the basic structure of MPLOT. A brief description of the 51 routines is provided in Section 4.3. Section 4.4 discusses subprogram calling structure and common block usage. Finally, files used by the program are described in Section 4.5.

4.2 MPLOT Program Structure

This section describes the high-level structure of the MPLOT program. Section 4.2.1 describes the main program and selection of plots. Section 4.2.2 describes the various plots and associated plot programs.

4.2.1 The MPLOT Main Program

The MPLOT main routine controls the selection of an input file, an output device and plots to be generated. Seven types of plots are currently supported by MPLOT:

1. Phase-by-Phase MCSP versus Time (PPPLT).

2. Critical Failure Rate versus Operating Time Since Repair (BCFPLT).

3. Pool MCSP Chain Budget (Series Chain) versus Pool Number (SERPLT).

DI-3000 is a trademark of Precision Visuals.

4. Chain MCSP Budget versus Chain Number (CHNPLT).

5. Pool MCSP Budget (Parallel Chain) versus Pool Type (PARPLT).

6. Relative Contribution to MCSP versus LRU/LRM (LRUPLT).

7. MCSP and Availability versus Repair Policy (REPPLT).

Figure 17 shows a tree diagram describing the MPLOT main routine and major subroutines called. Each main branch corresponds to one of the previously mentioned plot types. Figure 18 is a flowchart of the MPLOT program logic. Note that even though MIREM supplies MPLOT with MBTFF data, this plot type is not supported.

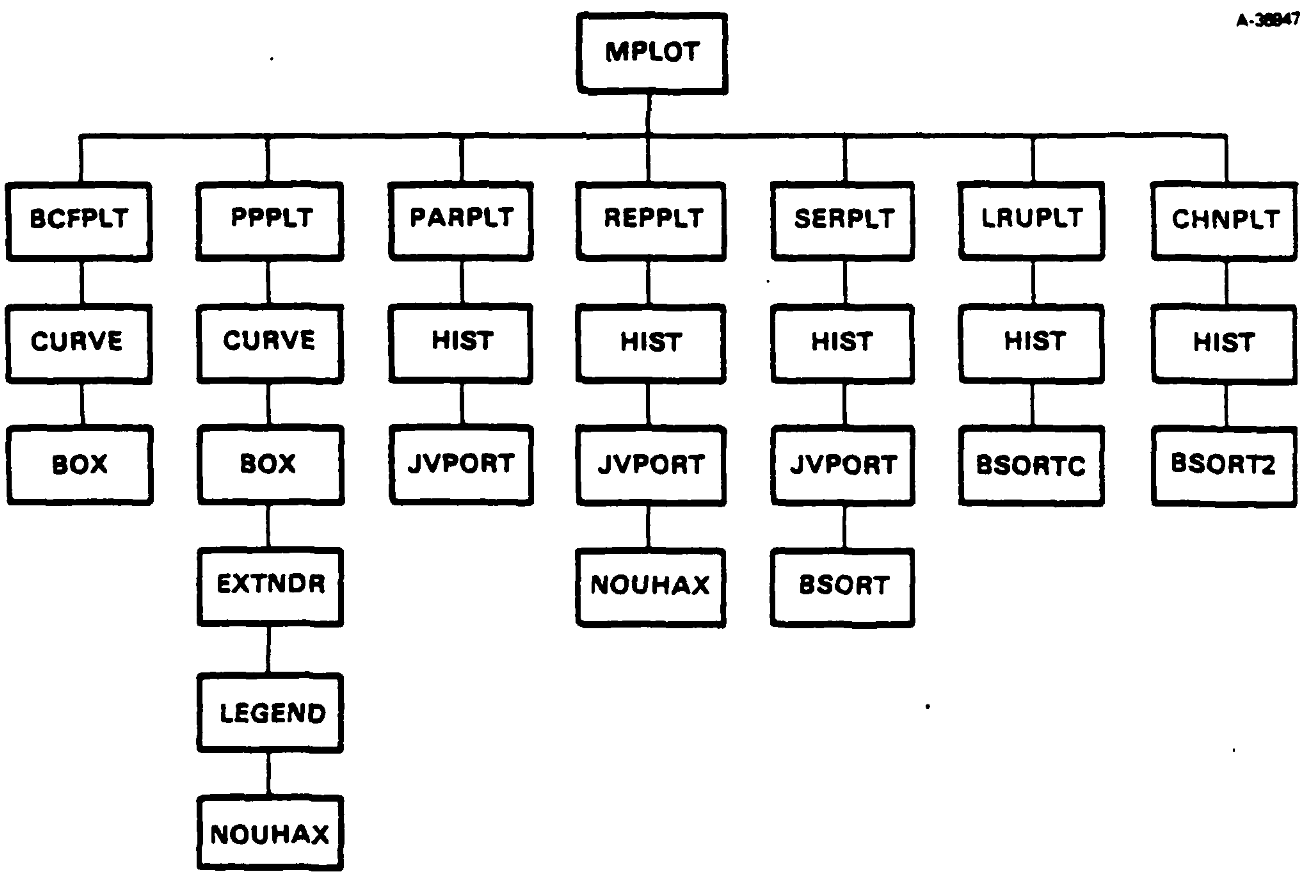


Figure 17. MPLOT Tree Diagram - Main Program.

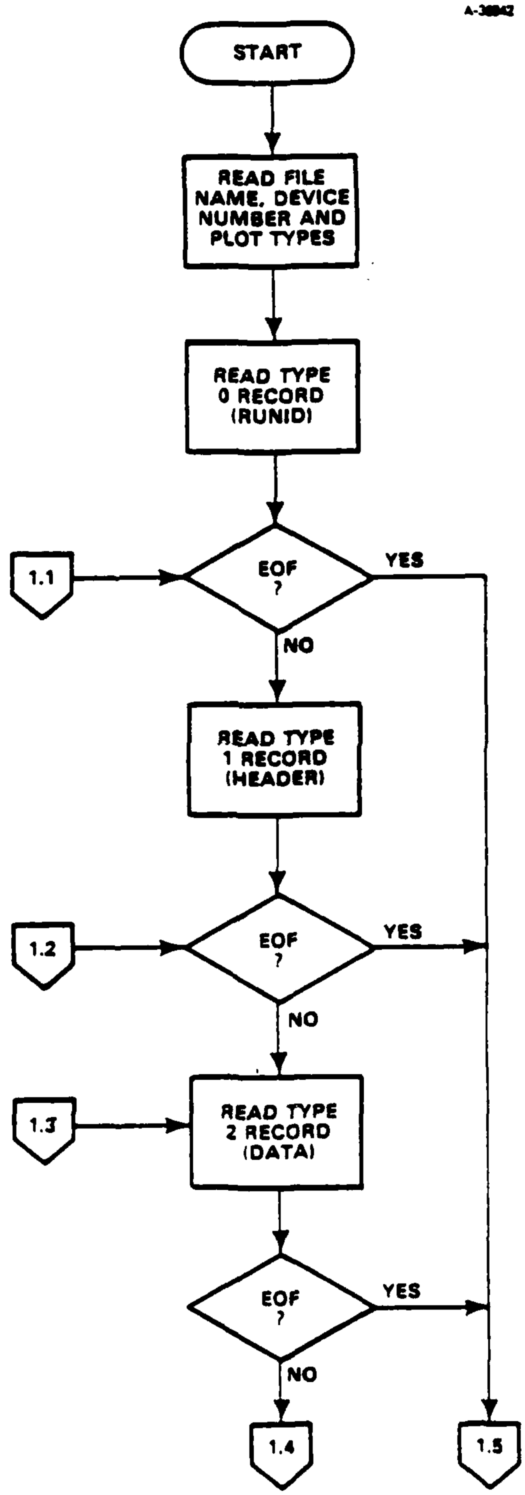


Figure 18. MPLOT Flowchart.

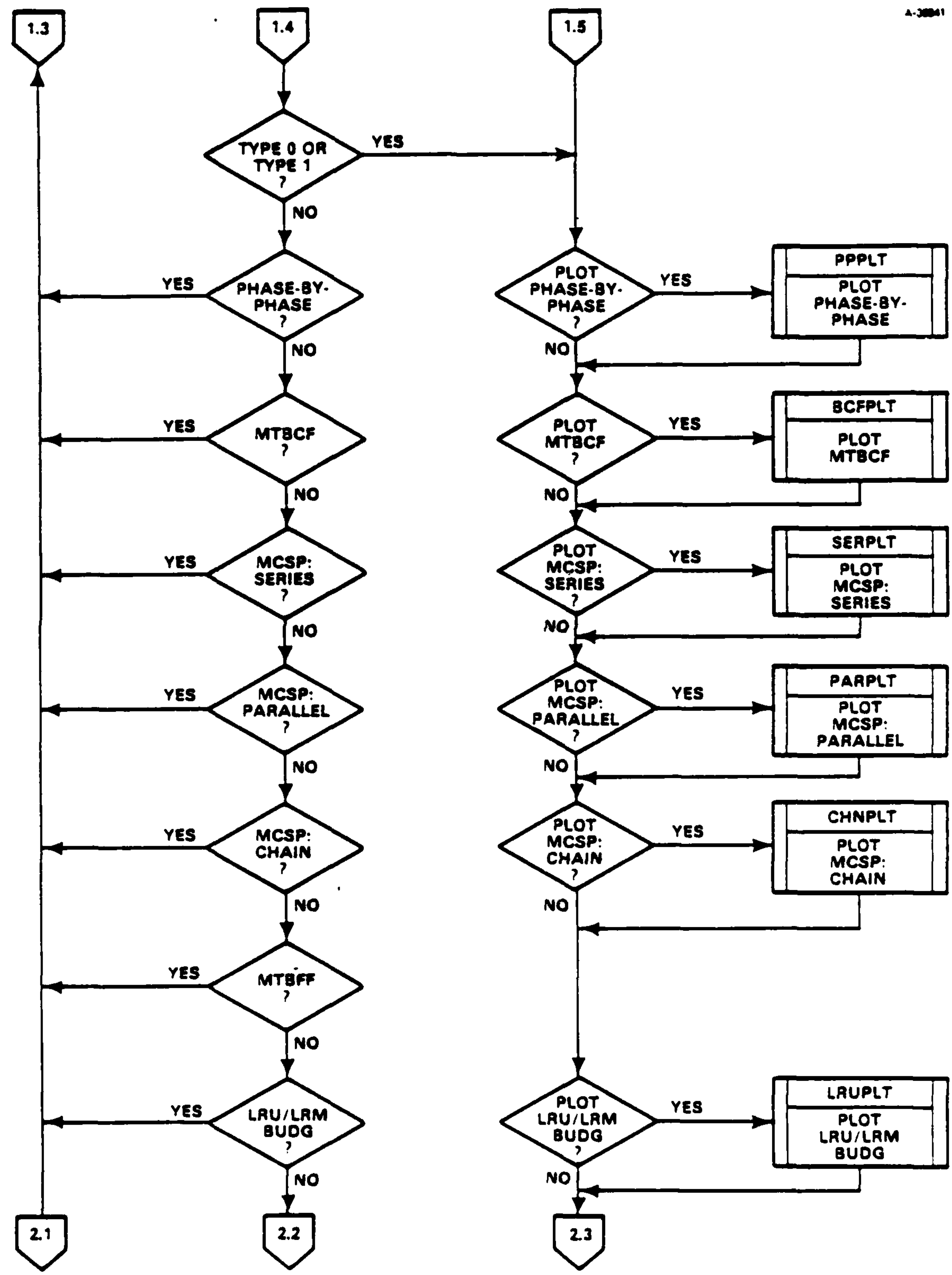


Figure 18. (Continued)

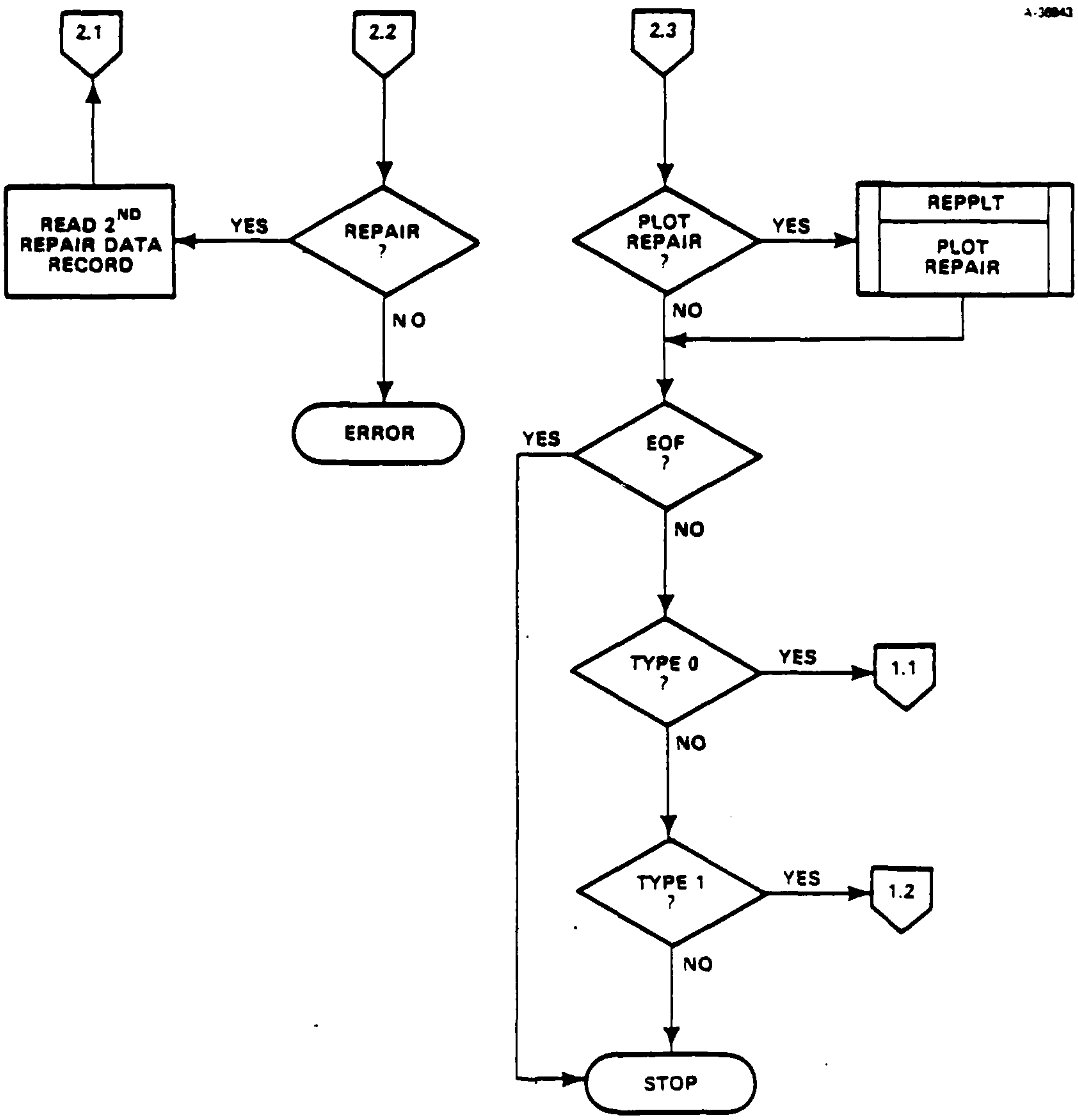


Figure 18. (Concluded)

4.2.2 The Main Plot Routines

There are seven routines for the seven plots in the MPLOT package. Each routine employs the same underlying logic and varies only slightly in function. All routines draw axes, scale the data, open and close plot segments, erase the screen before and after the plot, and set up appropriate windows and viewports. Table 11 shows the major differences and similarities in each of the routines.

Table 11. Features of MPLOT Routines

Routine	Type		Attributes					Data Manipulation
	Curve	Histogram	Boxed[a]	Extenders[b]	Legend[c]	3rd Axis[d]	Altered Viewport[e]	Sorted[f]
PPPLT	X		X	X	X	X	X	
BCFPLT	X		X					
PARPLT		X					X	
REPPLT		X				X	X	
SERPLT		X					X	X
LRUPLT		X						X
CHNPLT		X						X

a. A box is drawn around the data.

b. Vertical lines extend from the top of the plot to the bottom at each x point.

c. A legend is on the right side of the plot.

d. A third axis is displayed on the plot.

e. The space occupied by the plot on the display is altered from a default.

f. The data are sorted prior to display.

4.3 MPLOT Subroutine Descriptions

The 51 subprograms used in the MPLOT program are briefly described in Table 12. They are presented in alphabetic order with the exception of the main routine, MPLOT, which appears first.

4.4 MPLOT Subprogram Interdependencies and Common Block Usage

This section identifies the interdependence of subprograms in MPLOT in terms of transfer of control and the usage of common storage. Table 13 lists, for each routine, the subprograms it calls, routines that call it, and common blocks used. The description of the various parameters in the common blocks are found in each routine utilizing the specific common block.

Table 12. MPLOT Routine Descriptions

Routine	Description
MPLOT	Reads selected file, produces selected plots, and outputs plots to selected device.
BCFPLT	Produces plot of Critical Failure Rate versus Operating Time Since Repair.
BOX	Draws a box around the data area window.
BSORT	Performs a bubble sort on data and associated integer index array.
BSORT2	Performs a bubble sort on data and two associated integer index arrays.
BSORTC	Performs a bubble sort on data and associated character index array.
CENTER	Centers a character string.
CHAR2I	Performs character-to-integer conversion.
CHAR2R	Performs character-to-real conversion.
CHAROR	Sets up character attributes; i.e., size, angle, plane, etc.
CHARSZ	Calculates three character sizes in World Coordinates (small, medium, and large).
CHNPLT	Produces plot of MCSP Budget versus Chain Number.
COLOR	Sets up color tables for color plotting.
CURVE	Produces a plot of shaded (or unshaded) polygons under a curve.
CUSHDN	Calculates a new minimum for a given range of values.
CUSHUP	Calculates a new maximum for a given range of values.
DEVOFF	Deselects and terminates a plotting device.
DEVON	Selects and initializes a plotting device.

Table 12. (Continued)

Routine	Description
EXTNDR	Plots vertical or horizontal extenders at a given position.
EXTRMA	Calculates the area of plot surface not written on.
FRAME	Puts a frame around the entire plot including the title and axes.
HIST	Produces a histogram or multihistogram plot of shaded bars.
I2CHAR	Performs integer-to-character conversion.
LABEL	Labels an axis.
LEGEND	Outputs a legend on the far right side of a plot.
LINHAX	Plots a linear horizontal axis with ticks and tick annotations.
LINVAX	Plots a linear vertical axis with ticks and tick annotations.
LJUST	Left-justifies a character string.
LRUPLT	Produces a plot of Relative Contribution to MCSP versus LRU/LRM.
LSTYLE	Chooses a line style attribute.
MXMN	Finds the maximum and minimum of data.
NONHAX	Plots a nonuniform horizontal axis with or without ticks and tick annotations.
PARPLT	Produces a plot of Pool MCSP Budget (Parallel Chain) versus Pool Type.
PLEND	Deinitializes plot device and ends plotting session.
PLINIT	Initializes plot device and starts plotting session.
PNEXT	Clears plot device for next plot (i.e., screen erase or form feed).

Table 12. (Concluded)

Routine	Description
PPPLT	Produces a plot of Phase-by-Phase MCSP versus Time.
R2CHAR	Performs real-to-character conversion.
RATNL	Finds rational spacing for tick marks over a given range.
REPPLT	Produces a plot of MCSP and Availability versus Repair Policy.
RJUST	Right-justifies a string.
SERPLT	Produces a plot of Pool MCSP Chain Budget (Series Chain) versus Pool Number.
SETLEG	Readjusts viewport to accommodate a legend on the far right-hand side.
SHADE	Sets up color tables and shading flag.
SIGNIF	Calculates an output format for a real number given a number of significant digits.
STRLEN	Finds the length of a string excluding blank leaders and trailers.
STRNGS	Outputs a series of strings at a given point.
TITLE	Titles a plot.
VDPORT	Sets up default viewport; i.e., device area occupied by a plot.
WINDOW	Sets up the window in world (user) coordinates for a plot device viewport.
WMXMN	Defines a second window to help distinguish location and direction of plot axes.

Table 13. MPLOT Subprogram Interdependencies and Common Block Usage

Routine	Routines Called	Calling Routines	Common Blocks
MPLOT	PLINIT, PLEND, PNEXT, VDPORT, CHAR2I, PPPLT, BCFPLT, LRUPLT, CHNPLT, SERPLT, PARPLT, REPPLT, UF1ERS†, UF1PAR†	None	None
BCFPLT*	MXMN, RATNL, WINDOW, PNEXT, R2CHAR, JOPEN, SHADE, CURVE, JPINTR, LINHAX, LINVAX, TITLE, BOX, FRAME, JCLOSE, JPAUSE, CENTER, WMXMN, CUSHUP, SIGNIF, CHAR2R, LABEL	MPLOT	GRAFIX
BOX*	JIWIND, JPOLGN	BCFPLT, PPPLT	None
BSORT	None	SERPLT	None
BSORT2	None	CHNPLT	None
BSORTC	None	LRUPLT	None
CENTER	None	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	
CHAR2I	None	MPLOT	None
CHAR2R	None	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	None
CHAROR*	JFONT, JSIZE, JPATH, JJUST, JBASE, JPLANE	LEGEND, LINHAX, LINVAX, NOUHAX, STRNGS, WINDOW	None
CHARSZ	None	WINDOW	GRAFIX

Table 13. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
CHNPLT*	MXMN, RATNL, WINDOW, PNEXT, R2CHAR, JOPEN, SHADE, HIST, JPINTR, LINHAX, LINVAX, TITLE, FRAME, JCLOSE, JPAUSE, CENTER, R2CHAR, BSORT2, CUSHUP, WMXMN, I2CHAR, LJUST, LABEL, SIGNIF, CHAR2R	MPLOT	GRAFIX
COLOR*	JCOTBL	SHADE	GRAFIX
CURVE*	JCOLOR, JMOVE, JPOLY, JPINTR, JPEDGE, JPIDEX, JPOLGN, LSTYLE	BCFPLT, PPPLT	GRAFIX
CUSHDN	None	PPPLT	None
CUSHUP	None	BCFPLT, CHNPLT, LRUPLT, PARPLT, REPPLT, SERPLT	None
DEVOFF*	JDEVOF, JDEND	PLEND	None
DEVON*	JDEVON, JDINIT	PLINIT	None
EXTNDR*	JMOVE, JDRAW	PPPLT	GRAFIX
EXTRMA	None	LABEL	GRAFIX
FRAME*	J4RGET, JCONVW, JPOLGN	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	GRAFIX
HIST*	JPINTR, JPIDEX, JPOLGN	CHNPLT, LRUPLT, PARPLT, REPPLT, SERPLT	GRAFIX
I2CHAR	None	CHNPLT, PARPLT, R2CHAR, SERPLT	None
LABEL*	JMOVE, STRNGS, EXTRMA	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	GRAFIX
LEGEND*	JMOVE, JHSTRNG, CHAROR, STRLEN	PPPLT	GRAFIX

Table 13. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
LINHAX*	JMOVE, JDRAW, JHSTRNG, CHAROR, LJUST, RJUST, STRLEN, LSTYLE	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, SERPLT	GRAFIX
LINVAX*	JMOVE, JDRAW, JHSTRNG, CHAROR, LJUST, RJUST, STRLEN, LSTYLE	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	GRAFIX
LJUST	None	CHNPLT, LINHAX, LINVAX, LRUPLT, PARPLT, SERPLT	None
LRUPLT*	MXMN, RATNL, WINDOW, PNEXT, R2CHAR, JOPEN, SHADE, HIST, JPINTR, LINHAX, LINVAX, TITLE, FRAME, JCLOSE, JPAUSE, CENTER, CUSHUP, BSORTC, WMXMN, LJUST, LABEL, SIGNIF, CHAR2R, PNEXT	MPLOT	GRAFIX
LSTYLE*	JLSTYL	CURVE, LINVAX, LINHAX, NOUHAX, PPPLT	None
MXMN	None	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	None
NOUHAX*	JMOVE, JDRAW, JHSTRG, CHAROR, STRLEN, LSTYLE	PPPLT, REPPLT	GRAFIX
PARPLT*	MXMN, RATNL, WINDOW, PNEXT, R2CHAR, JOPEN, SHADE, HIST, JPINTR, LINHAX, LINVAX, TITLE, FRAME, JCLOSE, JPAUSE, CENTER, I2CHAR, LJUST, CUSHUP, JVPORT, PNEXT, WMXMN, LABEL, CHAR2R	MPLOT	GRAFIX
PLEND*	DEVOFF, JEND	MPLOT	None
PLINIT*	DEVON, JBEGIN, JWCLIP, JIDISA	MPLOT	GRAFIX

Table 13. (Continued)

Routine	Routines Called	Calling Routines	Common Blocks
PNEXT*	JFRAME	BCFPLT, CHNPLT, LRUPLT, MPLOT, PARPLT, PPPLT, REPPLT, SERPLT	None
PPPLT*	MXMN, RATNL, WINDOW, PNEXT, R2CHAR, JOPEN, SHADE, CURVE, JPINTR, LINHAX, LINVAX, TITLE, BOX, FRAME, JCLOSE, JPAUSE, CENTER, CUSHDN, SETLEG, LSTYLE, EXTNDR, WMXMN, SIGNIF, CHAR2R, LABEL, LEGEND, NOUHAX, PNEXT	MPLOT	GRAFIX
R2CHAR	None	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	None
RATNL	None	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	None
REPPLT*	MXMN, RATNL, WINDOW, PNEXT, R2CHAR, JOPEN, SHADE, HIST, JPINTR, LINHAX, LINVAX, TITLE, FRAME, JCLOSE, JPAUSE, CENTER, CUSHUP, JVPORT, WMXMN, NOUHAX, CHAR2R, LABEL, SIGNIF	MPLOT	GRAFIX
RJUST	None	LINHAX, LINVAX	None
SERPLT*	MXMN, RATNL, WINDOW, PNEXT, R2CHAR, JOPEN, SHADE, HIST, JPINTR, LINHAX, LINVAX, TITLE, FRAME, JCLOSE, JPAUSE, CENTER, CUSHUP, I2CHAR, LJUST, BSORT, JVPORT, WMXMN, LABEL, CHAR2R	MPLOT	GRAFIX
SETLEG	None	PPPLT	GRAFIX

Table 13. (Concluded)

Routine	Routines Called	Calling Routines	Common Blocks
SHADE*	JPINTR, JIQDEV, COLOR	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	GRAFIX
SIGNIF	None	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	None
STRLEN	None	LEGEND, LINHAX, LINVAX, NOUHAX	None
STRNGS*	JMOVE, JCP, JHSTRNG, CHAROR	LABEL, TITLE	GRAFIX
TITLE*	JMOVE, STRNGS	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	GRAFIX
VDPORT*	JASPEK, JVSPAC, JVPORT	MPLOT	GRAFIX
WINDOW*	CHARSZ, JWINDO	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	GRAFIX
WMXMN	None	BCFPLT, CHNPLT, LRUPLT, PARPLT, PPPLT, REPPLT, SERPLT	GRAFIX

*All routines beginning with "J" are DI-3000 routines.

†See DATAIN subroutine list.

4.5 MPLOT File Usage

Table 14 shows the logical device number assignments used by MPLOT for the reading and writing of files. All MPLOT sessions require terminal input specifying file name, device type, and plot selection.

Table 14. MPLOT Logical Device Assignments

Logical Device Number	File	Type	Format
5	Terminal Input	Input	Formatted
6	Terminal Output	Output	Formatted
10	MIREM Plot File	Input	Unformatted

The input plot file supplied by MIREM contains three general types of data. RUNID records (type 0) contain a 72-character string with the current mission run identifier. Header records (type 1) contain a title and either the total operating time or the MTBF of the data set. The title points to a specific format for the data records (type 2). Each header record is followed by one or more data records of format determined by the header record title. Tables 15, 16, and 17 show the format for each of these record types. Note that every record, regardless of type, begins with a word describing the number of words in the record, followed by a word describing the record type. Also note that there are always two 'REPAIR' records, each with its own format: MCSP, which always occurs first, and Availability.

Table 15. Plot File Format - RUNID (Type 0)

Word	Description	Declared Type
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=0: The RUNID record type	INTEGER
2-NWORDS	RUNID: The mission run identifier	CHARACTER*72

Table 16. Plot File Format - Header (Type 1)

Word	Description	Declared Type
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=1: The header record type	INTEGER
2	T or MTBF: The total operating time or Mean Time Between Failure, depending on type of data	REAL
3-NWORDS	TITLE: Data in following records is of type: TITLE — T/MTBF 'Phase-by-Pha' — N/A 'MTBCF' — MTBF 'MCSP: SERIES' — T 'MCSP: PARALLE' — T 'MCSP: CHAIN' — T 'MTBFF' — MTBF 'LRU/LRM BUDG' — T 'REPAIR' — T	CHARACTER*12

Table 17. Plot File Formats - Data (Type 2)

Word	Description	Declared Type
	'Phase-by-Pha'	
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=2: The data record type	INTEGER
2	TIME	REAL
3-4	MCSP lower bound	DOUBLE PRECISION
5-6	MCSP upper bound	DOUBLE PRECISION
7-NWORDS	Phase title	CHARACTER*32

Table 17. (Continued)

Word	Description	Declared Type
	'MTBCF'	
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=2: The data record type	INTEGER
2	TIME	REAL
3	Failure Rate	REAL
	'MCSP: SERIES'	
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=2: The data record type	INTEGER
2	Chain Number	INTEGER
3	Pool Number	INTEGER
4-5	Pool Probability	DOUBLE PRECISION
	'MCSP: PARALLE'	
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=2: The data record type	INTEGER
2	Primary Chain Number	INTEGER
3	Secondary Chain Number	INTEGER
4-5	Probability type 'N' Pools ok (for full P('N')=-2.0 and P('C')=P('N' or 'C'))	DOUBLE PRECISION
6-7	Probability type 'C' Pools ok	DOUBLE PRECISION
8-9	Probability type 'S' Pools ok	DOUBLE PRECISION
10-11	Probability type 'F' Pools ok	DOUBLE PRECISION
12-13	Chain Probability	DOUBLE PRECISION
	'MCSP: CHAIN'	
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=2: The data record type	INTEGER
2	Primary Chain Number	INTEGER
3	Secondary Chain Number (0 if series)	INTEGER
4-5	Chain Probability	DOUBLE PRECISION
6-NWORDS	Chain Name	CHARACTER*32

Table 17. (Concluded)

Word	Description	Declared Type
	'MTBFF'	
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=2: The data record type	INTEGER
2	MTBCF/Deferred Repair	REAL
3-NWORDS	Function Name	CHARACTER*8
	'LRU/LRM BUDG'	
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=2: The data record type	INTEGER
2-3	QM: Relative cumulative contribution of a particular LRU	DOUBLE PRECISION
4-NWORDS	LRU Name	CHARACTER*16
	'REPAIR' - MCSP	
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=2: The data record type	INTEGER
2-3	MCSP - Immediate Repair	DOUBLE PRECISION
4-5	MCSP - Deferred Repair	DOUBLE PRECISION
6-7	MCSP - Scheduled Maintenance	DOUBLE PRECISION
8-9	MCSP - Degraded Level	DOUBLE PRECISION
	'REPAIR' - AVAILABILITY	
0	NWORDS: The number of integer words in the record	INTEGER
1	TYPE=2: The data record type	INTEGER
2-3	Availability - Immediate Repair	DOUBLE PRECISION
4-5	Availability - Deferred Repair	DOUBLE PRECISION
6-7	Availability - Scheduled Maintenance	DOUBLE PRECISION
8-9	Availability - Degraded Level	DOUBLE PRECISION

5. PORTABILITY CONSIDERATIONS

The MIREM program was developed on TASC's IBM model 4381 computer. The DATAIN and MPLOT programs were developed on TASC's VAX 11/782 computer. All programs were targeted for delivery to the USAF Avionics Laboratory's VAX 11/780. The programs are expected to have a long life cycle and may be ported to other computers in the future. Although hardware and software differences exclude complete portability of code, much was done in the programs to minimize these problems. The major issues of portability addressed during software development include:

1. Choice of language.
2. Modularity of design to simplify modifications.
3. Storage requirements.
4. Computer terminal characteristics.
5. Simplicity of input and output files.
6. Use of mixed case alphabet.
7. Assumptions about machine word length.
8. Isolation and documentation of machine dependencies.

The DATAIN and MIREM programs were written using only ANSII standard FORTRAN 77. This language is widely accepted in the industry and provides efficient and readable programs. No vendor-specific extensions to the language were used. The MPLOT program uses one widely available extension to FORTRAN 77, as discussed below.

The use of modular design by all programs will simplify future program modifications. The isolation of different types of processing to different routines means that program maintenance is isolated to only those routines affected by a change. In addition, new features can be added as separate routines utilizing much of the existing program structure.

In order to run on machines with smaller internal memory, storage management must be considered. For this reason, the

HELP screens used by the user interface are kept on a file separate from the program. The modularity of the design also helps in storage management by isolating often-used computations to particular often-called subroutines. Moreover, modularity helps when machine internal memory sizes are so small that program overlays are required.

The programs are being accessed from both IBM 3270 and DEC VT-100 terminals as well as third-party terminals, requiring the DATAIN program to be largely terminal-independent. The user interface was designed to operate on any terminal with a width of at least 80 characters. These screens were tailored to look best on a 24-line CRT terminal, but this is not a requirement.

The files accessed were simplified for easy hosting in any environment. Both DATAIN and MIREM read one character at a time on input, which is important for portability to some microprocessors. The terminal output is written one line at a time and does not assume a full-screen interface. All files are card-image files with 80-character records. These are read sequentially; no direct access is used. The I/O is all executed under format control, so the character type may be ASCII or BCD or EBCDIC. MPLOT uses a standard unformatted direct-access read of binary data.

The program DATAIN makes heavy use of mixed-case character strings. Some compilers will not accept lowercase characters so these strings must be changed if this is the case.

The programs make no assumptions about the underlying machine word length. All character string lengths are explicitly given in the CHARACTER parameter declarations. All real, double-precision, and integer parameters use the length assigned by default by the compiler. This allows portability to machines with word lengths very different from the 32-bit IBM and VAX word lengths, such as the 60-bit word CDC machines. Note that the MIREM and MPLOT programs require a precision of at least six or seven decimal digits because of numerical considerations.

Another benefit of the above discussed modularity is the isolation of the machine-dependent code. The transition from the IBM to the VAX required slight changes to only four routines; namely, routines SCRID, UF1ERS, and UF1OPN in the DATAIN program and the routine HEADER in the MIREM program.

The routines SCRID in DATAIN and HEADER in MIREM contain calls to subroutines DATE and TIME. These DATE and TIME routines are intrinsic to the VAX system, returning a nine-character date in a 'DD-MMM-YY' format and an eight-character time in an 'HH:MM:SS' format, respectively. The IBM version

includes a similar DATE and TIME routine, utilizing machine-dependent code. Alternatively, the SCRID and HEADER routines could be modified to conform with another computer system's version of the DATE and TIME routines.

The subroutine UF1ERS in DATAIN and MPLOT contains a call to the VAX system routine LIB$ERASE_PAGE to clear the terminal screen. This call must be changed on other machines to enable the program to perform the terminal clearing. Additionally, the call must be updated if any third-party terminals (e.g., a Tektronix graphics terminal) are used.

The subroutine UF1OPN in DATAIN opens files using VAX-compatible file names "HELP.DAT" and "SCREEN.DAT". These default file names may need to be made compatible with use on another computer.

The MPLOT program requires a few special considerations. It is designed to interface with DI-3000 and cannot be run without an explicit link to the DI-3000 library. DI-3000 is a GKS-like (Graphics Kernel System) package; therefore, MPLOT graphics calls could be easily altered to accommodate another GKS-like graphics package.

Finally, a software technique was employed in MPLOT that is not standard FORTRAN 77. ANSII standard FORTRAN 77 does not permit character and binary data to occupy the same location in memory. This means that an EQUIVALENCE statement assigning both character and binary data to the same variable is invalid. The VAX and IBM both allow this extension to the language. This use of an EQUIVALENCE statement is an integral part of the main routine MPLOT. It is used when reading a record of MIREM plot file to make the transitions between type differences easier and more understandable.

REFERENCES

1. Veatch, M.H. (1986). Mission Reliability Model Users Guide (AFHRL-TR-86-35). Wright-Patterson Air Force Base, OH: Logistics and Human Factors Division, Air Force Human Resources Laboratory.

APPENDIX A: SAMPLE SCENARIO AND ARCHITECTURE FILES

Figure A-1 shows a sample Scenario file and Figure A-2 shows an Architecture file.

```
RUNID TEST CASE NUMBER 1
*********************************************************************
COMPUTE             MCSP PHASE LRU MTBCF MTBFF REPAIR BIT FULLBIT
PLOT                MCSP MTBCF PHASE REPAIR
QUICK               YES
PRINT HARDWARE      NO
PRINT INTERMEDIATE  NO
SIMULTANEOUS        YES
SCALE                  1.0
TIME                 446.    HOURS
TMAINTENANCE         446.    HOURS
REPSEQUENCE         PARALLEL
*********************************************************************
* PHASE CARD:           PHASE  TIME-OF-PHASE   PHASE
*                       INDEX     (HOURS)      NAME
*********************************************************************
* PHASE FUNCTION CARD:  PHASE
*                       INDEX                  LIST OF FUNCTION INDICES
*********************************************************************
PHASE                     1        1.50        PHASE 1
PHASE FUNCTION            1                    2
PHASE                     2        1.00        PHASE 2
PHASE FUNCTION            2                    1 3
PHASE                     3        0.50        PHASE 3
PHASE FUNCTION            3                    2
```

Figure A-1. Sample Scenario File.

```
*********************************************************************
* FUNCTION CARD: LIST OF FUNCTION NAMES
*********************************************************************
FUNCTION  'GPS     ' 'UHF     ' 'SINC    '
*********************************************************************
* LRU CARD: LIST OF LRM/LRU NAMES
*********************************************************************
LRU        'FRONTEND         ' 'DIGITALA          ' 'DIGITALB        '
*********************************************************************
* RESOURCE : RESOURCE QUAN- FAILURE TYPE       RESOURCE
*     CARD :  NUMBER   TITY RATE(1) (2)  MTTR NAME
*********************************************************************
RESOURCE          1       1      10     R  4.0  L-BAND ANTENNA CONNECTOR
RESOURCE          2       2      15     R  3.5  L-BAND RECEIVER
RESOURCE          3       2       5     R  2.0  2 X 3 L-BAND SWITCH PORTS
RESOURCE          4       1      10     R  2.5  LOW-BAND ANTENNA SWITCH
RESOURCE          5       2      95     R  4.0  LOW-BAND RECEIVER
RESOURCE          6       2       5     R  3.0  2 X 5 LOW-BAND SWITCH PORTS
RESOURCE          7       5     300     R  2.0  PREPROCESSOR
RESOURCE          8       2     100     R  5.0  SIGNAL PROCESSOR
RESOURCE          9       2      20     R  4.5  POWER SUPPLY
RESOURCE         10       2      20     R  4.0  SDU I/O
RESOURCE         11       2     100     R  6.0  CONTROLLER
*********************************************************************
* (1) NOTE: FAILURE RATE IS IN PER MILLION HOURS
* (2) NOTE: 'R' IS RESOURCE, 'I' IS INTERCONNECTION
*********************************************************************
* CHAIN CARD:         PRIMARY     PARALLEL
*                     CHAIN       CHAIN       CHAIN NAME
*                     NUMBER      NUMBER(3)   NAME
*********************************************************************
* CHAIN FUNCTION : CHAIN
*           CARD : NUMBER                     LIST OF FUNCTION INDICES
*********************************************************************
CHAIN                  1           0          FRONT END
CHAIN FUNCTION         1                      1  2  3
CHAIN                  2           3          DIGITAL
CHAIN FUNCTION         2                      1  2  3
CHAIN FUNCTION         3                      2  3
*********************************************************************
* (3) NOTE: PARALLEL CHAIN NUMBER IS 0 WHEN CHAIN IS SERIES CHAIN
```

Figure A-2. Sample Architecture File.

```
****************************************************************
* POOL CARD   :                     POOL NO.   ACTIVE/ UNDET.   FALSE MINIMUM
*             : POOL CHAIN LRU      TYPE BRAN- STANDBY FAILURE  ALARM REPAIR
*             : NO.  NO.   INDEX    (4)  CHES  (5)     RATE     RATE  LEVEL
****************************************************************
* POOL        :
* RESOURCE    : POOL
* CARD        : NO.                            LIST OF RESOURCE NUMBERS
****************************************************************
* POOL        :
* UTILIZATION : POOL
* CARD        : NO.                            LIST OF FUNCTION UTILIZATIONS
****************************************************************
POOL            10    1    1    N    1    A       0.1      0.05   1
POOL RESO       10                        1  2 2  3 3
POOL UTIL       10                        1.00  0.00  0.00
POOL            11    1    1    N    1    A       0.005    0.001  1
POOL RESO       11                        4
POOL UTIL       11                        0.00  1.00  1.00
POOL            12    1    1    N    2    S       0.02     0.01   1
POOL RESO       12                        5  6
POOL UTIL       12                        0.00  1.00  1.00
POOL            13    2    2    C    3    A       0.01     0.01   2
POOL RESO       13                        7
POOL UTIL       13                        2.00  1.00  1.00
POOL            14    2    2    S    1    A       0.02     0.005  1
POOL RESO       14                        8
POOL UTIL       14                        0.80  0.10  0.40
POOL            15    2    2    F    1    A       0.       0.01   1
POOL RESO       15                        9
POOL UTIL       15                        1.00  1.00  1.00
POOL            16    2    2    N    1    A       0.01     0.005  1
POOL RESO       16                        10
POOL UTIL       16                        0.00  0.00  1.00
POOL            17    2    2    N    1    A       0.05     0.02   1
POOL RESO       17                        11
POOL UTIL       17                        1.00  1.00  1.00
POOL            13    3    3    C    2    A       0.01     0.01   2
POOL RESO       13                        7
POOL UTIL       13                        2.00  1.00  1.00
POOL            14    3    3    S    1    A       0.02     0.005  1
POOL RESO       14                        8
POOL UTIL       14                        0.80  0.10  0.40
POOL            15    3    3    F    1    A       0.       0.01   1
POOL RESO       15                        9
POOL UTIL       15                        1.00  1.00  1.00
POOL            16    3    3    N    1    A       0.01     0.005  1
POOL RESO       16                        10
POOL UTIL       16                        0.00  0.00  1.00
POOL            17    3    3    N    1    A       0.05     0.02   1
POOL RESO       17                        11
POOL UTIL       17                        1.00  1.00  1.00
****************************************************************
* (4) NOTE: 'N' IS NONCONTENDING, 'C' IS CONTENDING,
*           'S' IS SHARED,   AND  'F' IS CHAIN-FAIL
****************************************************************
* (5) NOTE: 'A' IS ACTIVE, 'S' IS STANDBY
```

Figure A-2. (Concluded)

APPENDIX B: SAMPLE MIREM OUTPUT REPORTS

Figure B-1 displays a sample MCSP and budget report: Figure B-2 a sample Phase-by-Phase MCSP Report; Figure B-3, a sample MTBCF Report; Figure B-4, a sample MTBFF Report; Figure B-5, a sample LRM/LRU Budget Report; Figure B-6, a sample Repair Policy Report; Figure B-7, a sample Testability Factors Report for the BIT Option; and Figure B-8, a sample Testability Factors Report for the Full BIT Option. Two other reports reflect the contents of the input files. Figure B-9 shows a sample Scenario File Report and Figure B-10, a sample Architecture File Report.

TEST CASE NUMBER 1

MCSP AND BUDGET OUTPUT OPTION

CHAIN NUMBER	CHAIN NAME	SERIES/PARALLEL
1	FRONT END	SERIES
	POOL NUMBER	POOL MCSP
	10	0.977947
	11	0.995550
	12	0.999035
	CHAIN MCSP:	0.972655
2,3	DIGITAL	PARALLEL
	TYPE N POOL MCSP:	0.955937
	TYPE C POOL MCSP:	0.983863
	TYPE S POOL MCSP:	0.914663
	TYPE F POOL MCSP:	0.982318
	CHAIN MCSP (BOTH CHAINS OPERABLE):	0.845040
	CHAIN MCSP (PRIMARY CHAIN ONLY):	0.000000
	CHAIN MCSP (SECONDARY CHAIN ONLY):	0.000000
	CHAIN MCSP:	0.845040
IMMEDIATE REPAIR MTBCF:		2274.
TOTAL SYSTEM MCSP AT TIME 446.00 HOURS:		0.821932

Figure B-1. Sample MCSP and Budget Report.

PHASE-BY-PHASE MCSP REPORT

TIME INTO MISSION	CURRENT PHASE	SUCCESS PROBABILITY LOWER BOUND	UPPER BOUND
0.00	START OF MISSION	1.000000	1.000000
1.50	PHASE 1	0.999985	0.999985
2.50	PHASE 2	0.998985	0.999585
3.00	PHASE 3	0.998955	0.999580
SYSTEM MCSP		0.998955	0.999580
IMMEDIATE REPAIR MTBCF(HOURS)		2870.	7140.

Figure B-2. Sample Phase-by-Phase MCSP Report.

MTBCF REPORT

MEAN TIME BETWEEN CRITICAL FAILURES (MTBCF): 1459. HOURS
MEAN TIME BETWEEN FAILURES (MTBF): 446. HOURS
FAILURE RESILIENCY: 3.27

INTEGRATION INTERVALS

MIDPOINT (HOURS)	WIDTH (HOURS)	FAILURE RATE (E-6/HOURS)	AREA (HOURS)
5.58	11.16	400.1	11.14
33.48	44.64	401.5	44.05
145.09	178.57	417.7	168.34
591.52	714.29	553.4	542.43
1143.58	389.84	733.5	208.65
1502.95	328.91	830.4	132.99
1744.00	153.20	885.9	50.14
1944.09	246.97	926.2	67.65
2182.62	230.09	969.1	50.27
2402.97	210.62	1004.2	37.01
2607.09	197.61	1033.2	28.19
2800.19	188.58	1058.1	21.98
2985.36	181.76	1079.8	17.38
3173.08	193.70	1100.0	15.11
3386.50	233.13	1120.8	14.38
3647.67	289.22	1143.8	13.34
3981.85	379.13	1169.5	12.00
4441.52	540.20	1199.5	10.18
5153.89	884.55	1236.4	7.64
6532.02	1871.70	1285.3	4.43
10758.80	6581.86	1350.4	1.29

NUMBER OF MCSP EVALUATIONS: 24
NUMBER OF INTERVALS: 21
STOPPING POINT: 14049.73 HOURS
FINAL MCSP: 0.000000
CONSTANT FAILURE RATE MTBCF: 2499. HOURS

Figure B-3. Sample MTBCF Report.

MTBFF REPORT

TOTAL OPERATING TIME: 446.00 HOURS

FUNCTION	MCSP	MTBCF IMMEDIATE REPAIR	MTBCF DEFERRED REPAIR	FAILURE RESILIENCY
GPS	0.885321	3882.	1967.	4.41
UHF	0.988415	38275.	4224.	9.46
SINC	0.987350	35033.	4055.	9.08

Figure B-4. Sample MTBFF Report.

LRM/LRU BUDGET REPORT

TOTAL OPERATING TIME: 446.00 HOURS
MISSION DURATION: 3.00 HOURS

LRM/LRU	MARGINAL MCSP (CUMULATIVE)	RELATIVE CONTRIBUTION (CUMULATIVE)	PROBABILITY OF REMOVAL UPON REPAIR
FRONTEND	0.845040	0.130	0.103812
DIGITALA	0.921967	0.562	0.588351
DIGITALB	0.880019	0.326	0.367499
INTERCONNECTIONS	N/A	N/A	N/A

SYSTEM MCSP (CUMULATIVE): 0.821932
SYSTEM MCSP (LAST MISSION): 0.998496

Figure B-5. Sample LRM/LRU Budget Report.

REPAIR POLICY REPORT

MISSION DURATION = 446.00 HOURS

QUANTITY	IMMEDIATE REPAIR	DEFERRED REPAIR	SCHEDULED MAINTENANCE	REPAIR AT DEGRADED LEVEL
AVERAGE MCSP	0.821932305	0.736550041	0.736015933	** N/A **
MTBCF	2274.	1459.	1455.	** N/A **
MTBMA	446.43	1458.57	553.21	745.35
MTTR	2.24	4.00	2.32	2.42
INHERENT AVAILABILITY	0.99501	0.99727	0.99583	0.99676

Figure B-6. Sample Repair Policy Report.

TEST CASE NUMBER 1

TESTABILITY FACTORS REPORT
BIT OPTION

MISSION DURATION = 446.00 HOURS

	LOWER BOUND	UPPER BOUND
PERFECT BIT MCSP	0.82193231	0.82193231
IMPERFECT BIT MCSP	0.72794329	0.72963352
PROBABILITY OF MISSION FAILURE DUE TO BIT	0.09398901	0.09229879
MISSION FAILURE FALSE ALARM PROBABILITY	0.00186934	0.00481517

Figure B-7. Sample Testability Factors Report - BIT Option.

TEST CASE NUMBER 1

TESTABILITY FACTORS REPORT
BIT MTBCF OPTION

MISSION DURATION = 446.00 HOURS

	LOWER BOUND	UPPER BOUND
MTBCF	1427.09	1436.09
MTBF	446.43	446.43
FAILURE RESILIENCY	3.20	3.22
MCSP AT TIME T	0.727943293	0.729633518

Figure B-8. Sample Testability Factors Report - Full Bit Option.

SCENARIO FILE REPORT

COMPUTATION/PLOT SELECTIONS:

1. DEFERRED REPAIR MTBCF (NO PLOT)
2. LRM/LRU BUDGET (NO PLOT)

NOTES:
MTBCF - MEAN TIME BETWEEN CRITICAL FAILURES

BASIC SCENARIO FILE PARAMETERS:

1.	PROCESSING OPTIONS:	QUICK
2.	PRINT HARDWARE FILE REPORT?:	YES
3.	PRINT INTERMEDIATE RESULTS?:	YES
4.	FUNCTIONS REQUIRED SIMULTANEOUSLY?:	YES
5.	FAILURE RATE SCALE FACTOR:	1.0
6.	TOTAL OPERATING TIME (HOURS):	446.00

MISSION PHASE LIST

INDEX	PHASE NAME	LENGTH (HOURS)	CRITICAL FUNCTIONS
1.	PHASE 1	1.50	2
2.	PHASE 2	1.00	1,3
3.	PHASE 3	0.50	2

Figure B-9. Sample Scenario File Report.

ARCHITECTURE FILE REPORT

POOL REPORT

CHAIN NUMBER	POOL NUMBER	LRM/LRU NAME	NUMBER BRANCHES	POOL FAILURE RATE *	POOL TYPE	REDUN-DANCY	MINIMUM LEVEL REPAIR	UNDETECTED FAILURE RATE	FALSE ALARM RATE	RESOURCE NUMBER	RESOURCE FAILURE RATE *	RESOURCE NAME
1	10	FRONTEND	1	50	NONCONTENDING	ACTIVE	1	0.100	0.050	1	10	L-BAND ANTENNA C
										2	15	L-BAND RECEIVER
										2	15	L-BAND RECEIVER
										3	5	2 X 3 L-BAND SWI
										3	5	2 X 3 L-BAND SWI
	11	FRONTEND	1	10	NONCONTENDING	ACTIVE	1	0.005	0.001	4	10	LOW-BAND ANTENNA
	12	FRONTEND	2	200	NONCONTENDING	STANDBY	1	0.020	0.010	5	95	LOW-BAND RECEIVE
										6	5	2 X 5 LOW-BAND S
2	13	DIGITALA	3	900	CONTENDING	ACTIVE	2	0.010	0.010	7	300	PREPROCESSOR
	14	DIGITALA	1	100	SHARED	ACTIVE	1	0.020	0.005	8	100	SIGNAL PROCESSOR
	15	DIGITALA	1	20	CHAIN-FAIL	ACTIVE	1	0.000	0.010	9	20	POWER SUPPLY
	16	DIGITALA	1	20	NONCONTENDING	ACTIVE	1	0.010	0.005	10	20	SDU I/O
	17	DIGITALA	1	100	NONCONTENDING	ACTIVE	1	0.050	0.020	11	100	CONTROLLER
3	13	DIGITALB	2	600	CONTENDING	ACTIVE	2	0.010	0.010	7	300	PREPROCESSOR
	14	DIGITALB	1	100	SHARED	ACTIVE	1	0.020	0.005	8	100	SIGNAL PROCESSOR
	15	DIGITALB	1	20	CHAIN-FAIL	ACTIVE	1	0.000	0.010	9	20	POWER SUPPLY
	16	DIGITALB	1	20	NONCONTENDING	ACTIVE	1	0.010	0.005	10	20	SDU I/O
	17	DIGITALB	1	100	NONCONTENDING	ACTIVE	1	0.050	0.020	11	100	CONTROLLER
SYSTEM FAILURE RATE				2240								

(*) FAILURE RATE IN PER MILLION HOURS

Figure B-10. Sample Architecture File Report.

ARCHITECTURE FILE REPORT

FUNCTION LIST

INDEX	FUNCTION NAME
1	GPS
2	UHF
3	SINC

ARCHITECTURE FILE REPORT

RESOURCE LIST

RESOURCE NUMBER	QUANTITY	FAILURE RATE (X E-6 HRS.)	RESOURCE/ INTERCONNECTION	MTTR (HOURS)	RESOURCE NAME
1	1	10	RESOURCE	4.0	L-BAND ANTENNA CONNECTOR
2	2	15	RESOURCE	3.5	L-BAND RECEIVER
3	2	5	RESOURCE	2.0	2 X 3 L-BAND SWITCH PORTS
4	1	10	RESOURCE	2.5	LOW-BAND ANTENNA SWITCH
5	2	95	RESOURCE	4.0	LOW-BAND RECEIVER
6	2	5	RESOURCE	3.0	2 X 5 LOW-BAND SWITCH PORTS
7	5	300	RESOURCE	2.0	PREPROCESSOR
8	2	100	RESOURCE	5.0	SIGNAL PROCESSOR
9	2	20	RESOURCE	4.5	POWER SUPPLY
10	2	20	RESOURCE	4.0	SDU I/O
11	2	100	RESOURCE	6.0	CONTROLLER

ARCHITECTURE FILE REPORT

CHAIN LIST

CHAIN PAIR NAME	CHAIN NUMBER	FUNCTIONS REQUIRED
FRONT END	1	1,2,3
DIGITAL	2	1,2,3
	3	2,3

Figure B-10. (Continued)

ARCHITECTURE FILE REPORT

FUNCTION UTILIZATION BY POOL

CHAIN NUMBER	POOL NUMBER	FUNCTION GPS	UHF	SINC
1	10	1.00	0.00	0.00
	11	0.00	1.00	1.00
	12	0.00	1.00	1.00
2	13	2.00	1.00	1.00
	14	0.80	0.10	0.40
	15	1.00	1.00	1.00
	16	0.00	0.00	1.00
	17	1.00	1.00	1.00
3	13	2.00	1.00	1.00
	14	0.80	0.10	0.40
	15	1.00	1.00	1.00
	16	0.00	0.00	1.00
	17	1.00	1.00	1.00

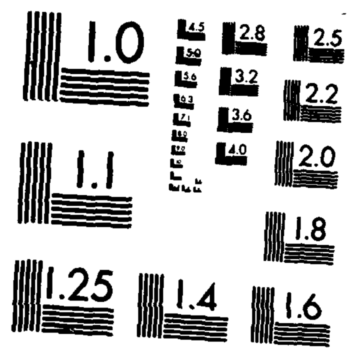
Figure B-10. (Concluded)

APPENDIX C: SAMPLE PLOTS

Figure C-1 shows a sample plot of Phase-by-Phase MCSP versus Time produced by routine PPPLT. Figure C-2 shows a sample plot of Critical Failure Rate versus Operation Time Since Repair produced by routine BCFPLT. Figure C-3 shows a sample plot of Pool MCSP Budget (Series Chain) versus Pool Number produced by routine SERPLT. Figure C-4 shows a sample plot of Chain MCSP Budget versus Chain Number produced by routine CHNPLT. Figure C-5 shows a sample plot of Pool MCSP Budget (Parallel Chain) versus Pool Type produce by routine PARPLT. Figure C-6 shows a sample plot of Relative Contribution to MCSP versus LRU/LRM produced by routine LRUPLT, and Figure C-7 shows a sample plot of MCSP and Availability versus Repair Policy produced by routine REPPLT.

AD-A174 760 MISSION RELIABILITY MODEL PROGRAMMERS GUIDE(U) ANALYTIC SCIENCES CORP READING MA J M MEDINA ET AL DEC 86 AFHRL-TP-86-38 F33615-82-C-0002 2/2

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

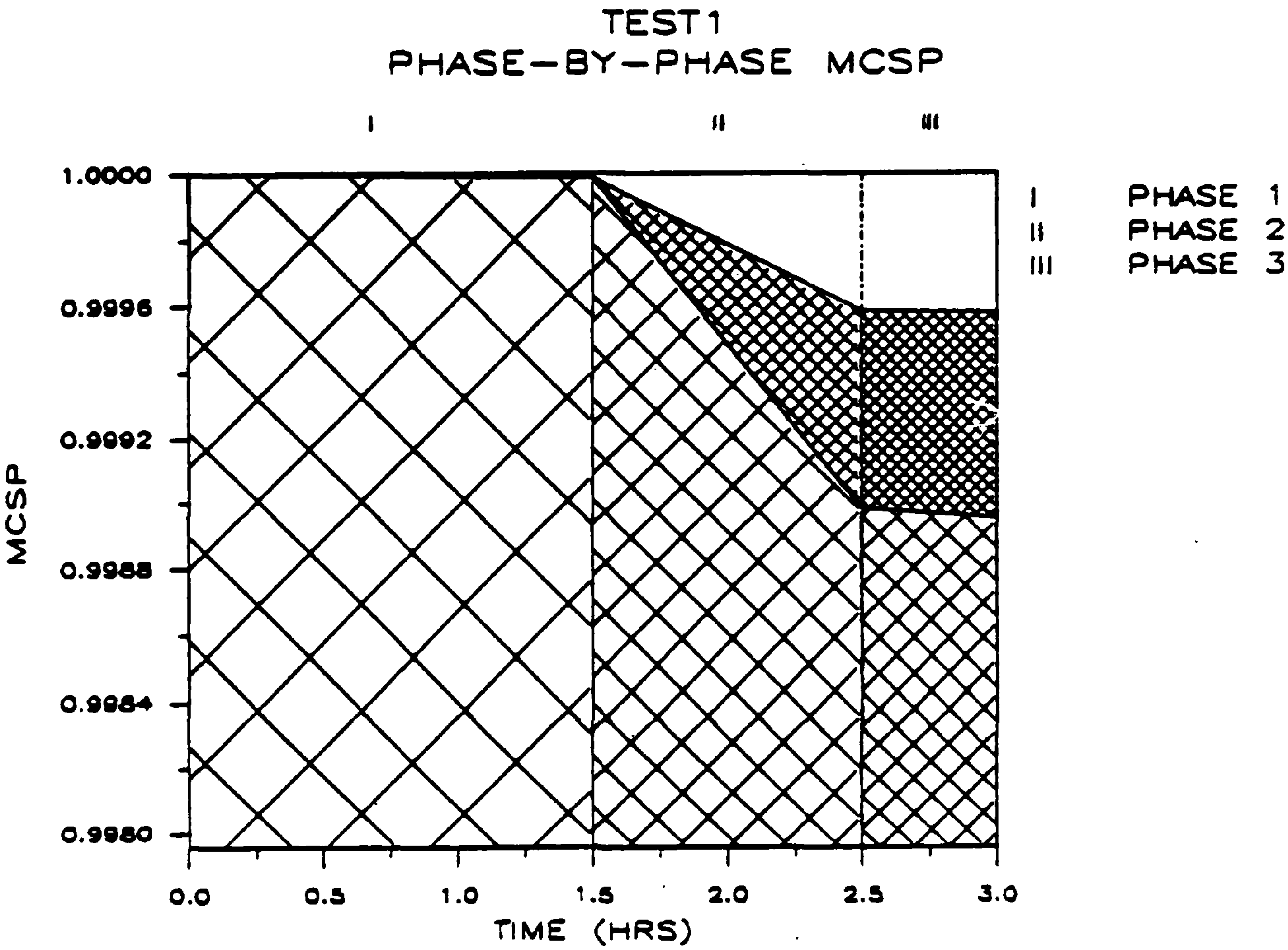


Figure C-1. Phase-by-Phase MCSP versus Time.

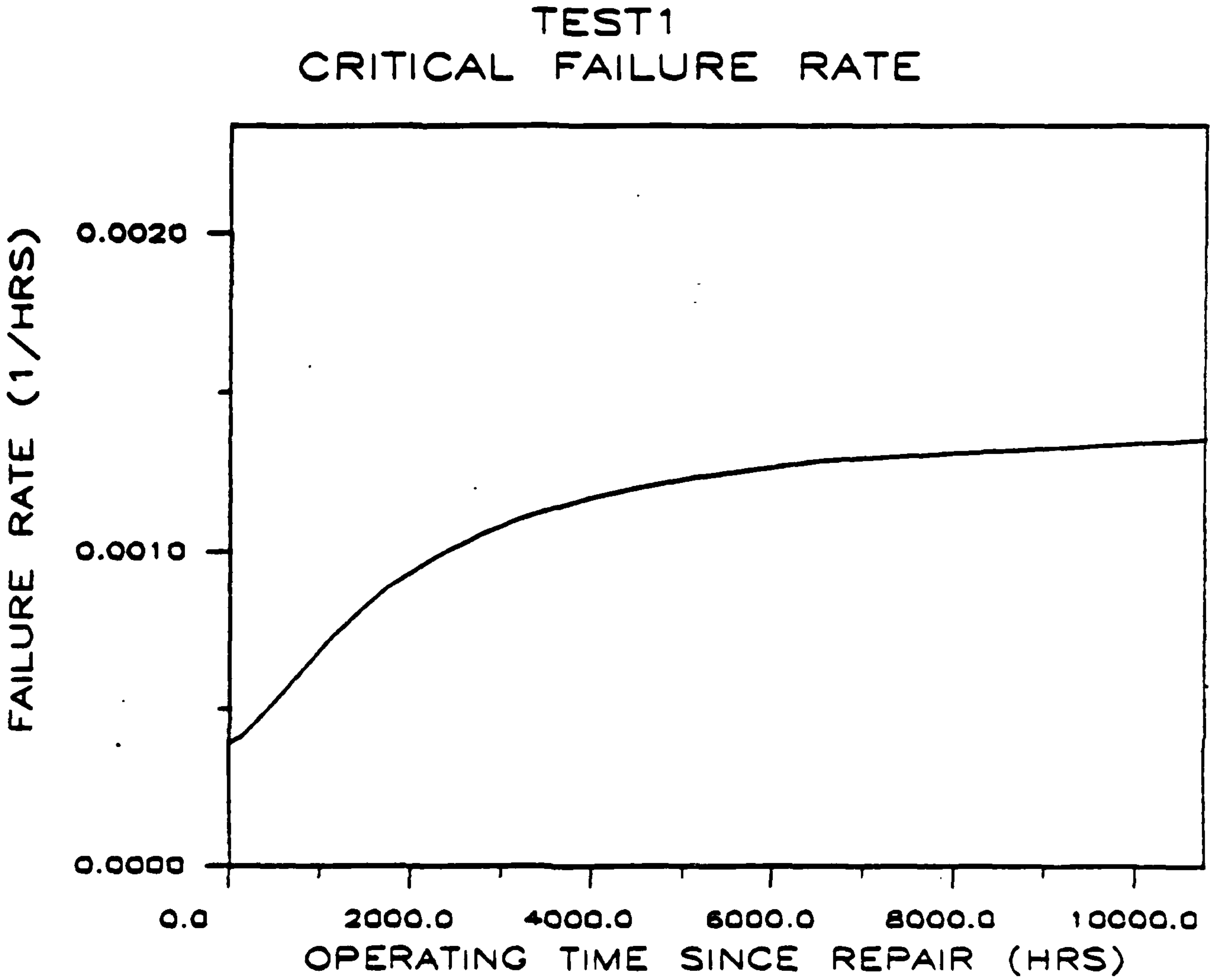


Figure C-2. Critical Failure Rate versus Operating Time Since Repair.

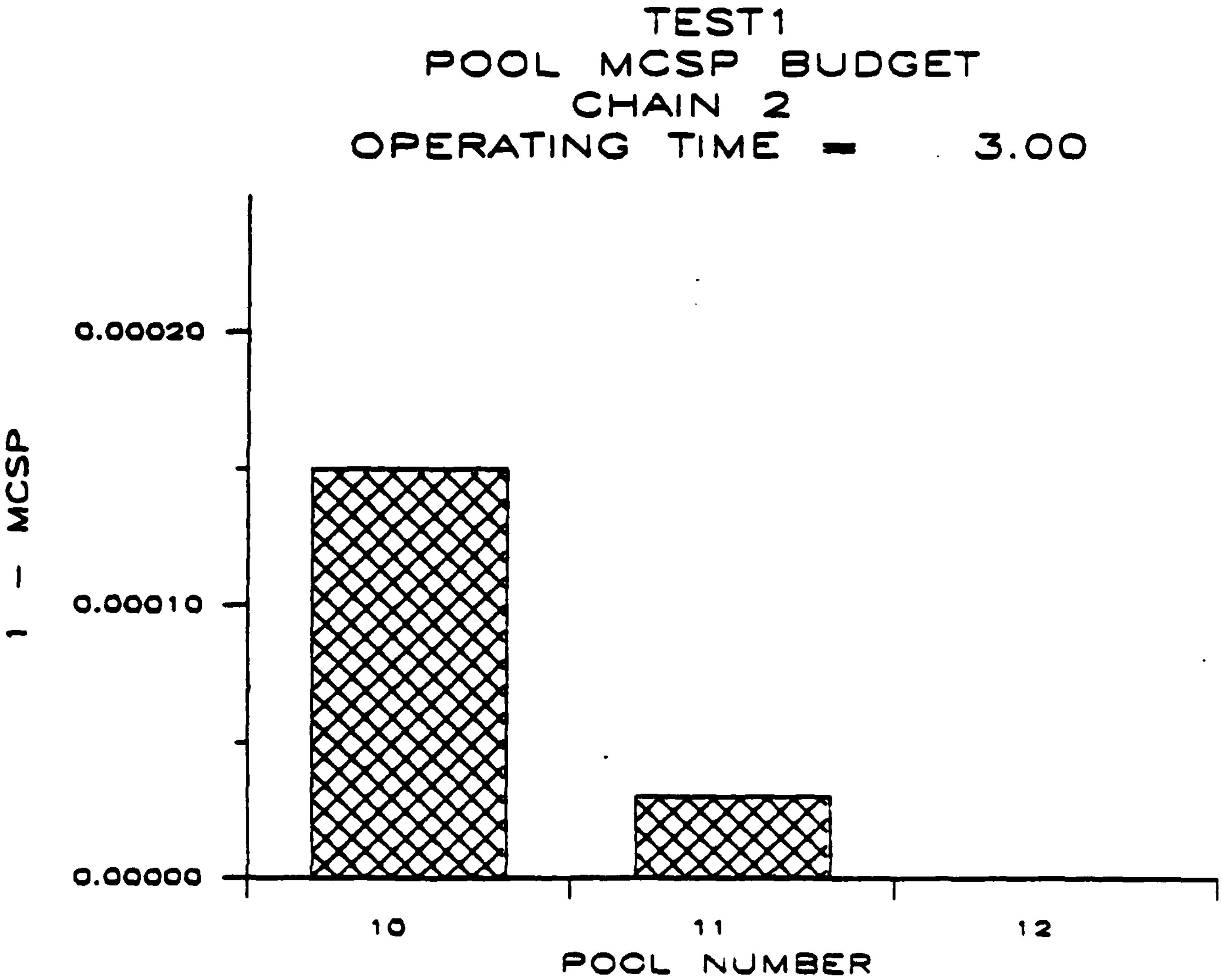


Figure C-3. Pool MCSP Budget (Series Chain) versus Pool Number.

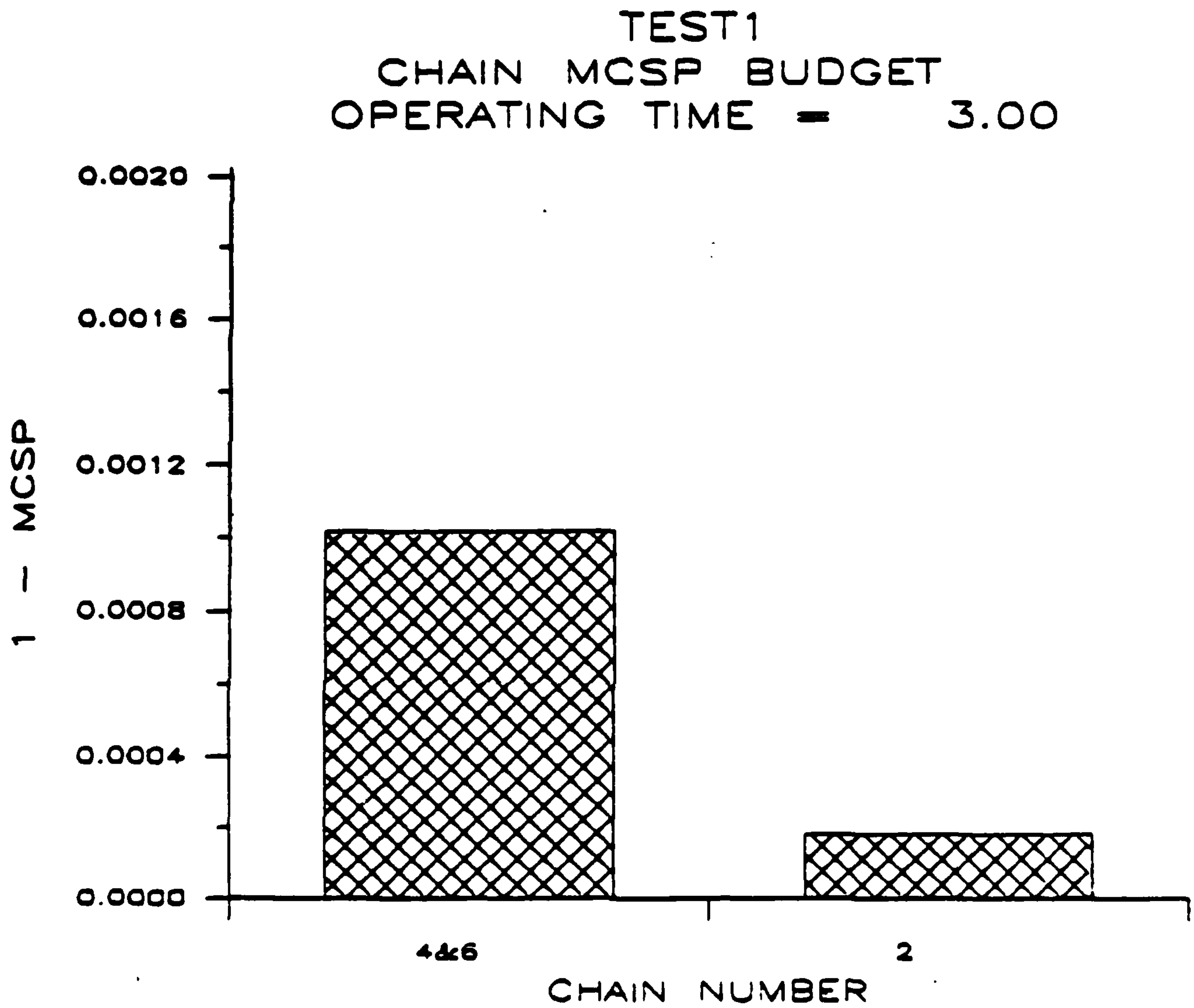


Figure C-4. Chain MCSP Budget versus Chain Number.

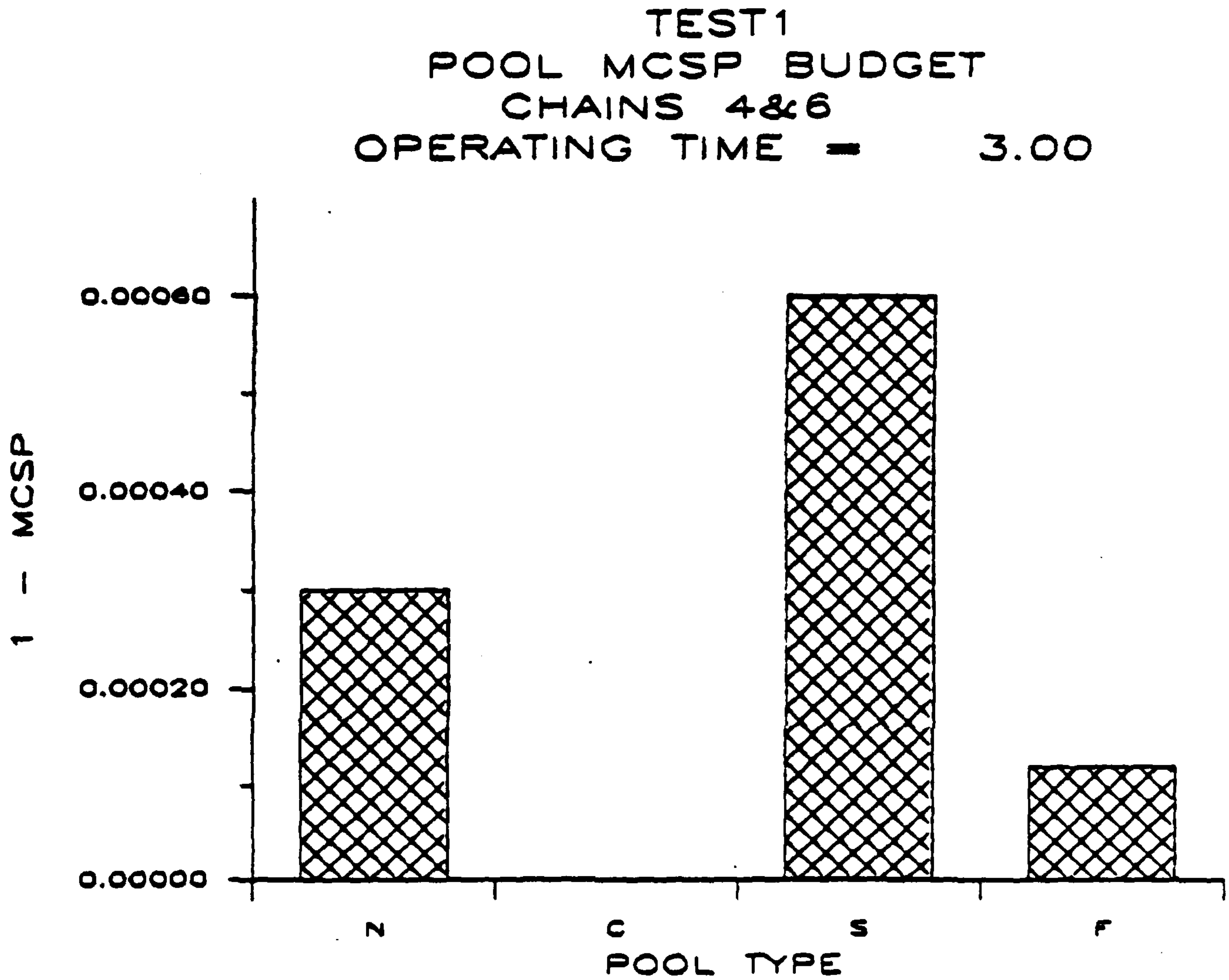


Figure C-5. Pool MCSP Budget (Parallel Chain) versus Pool Type.

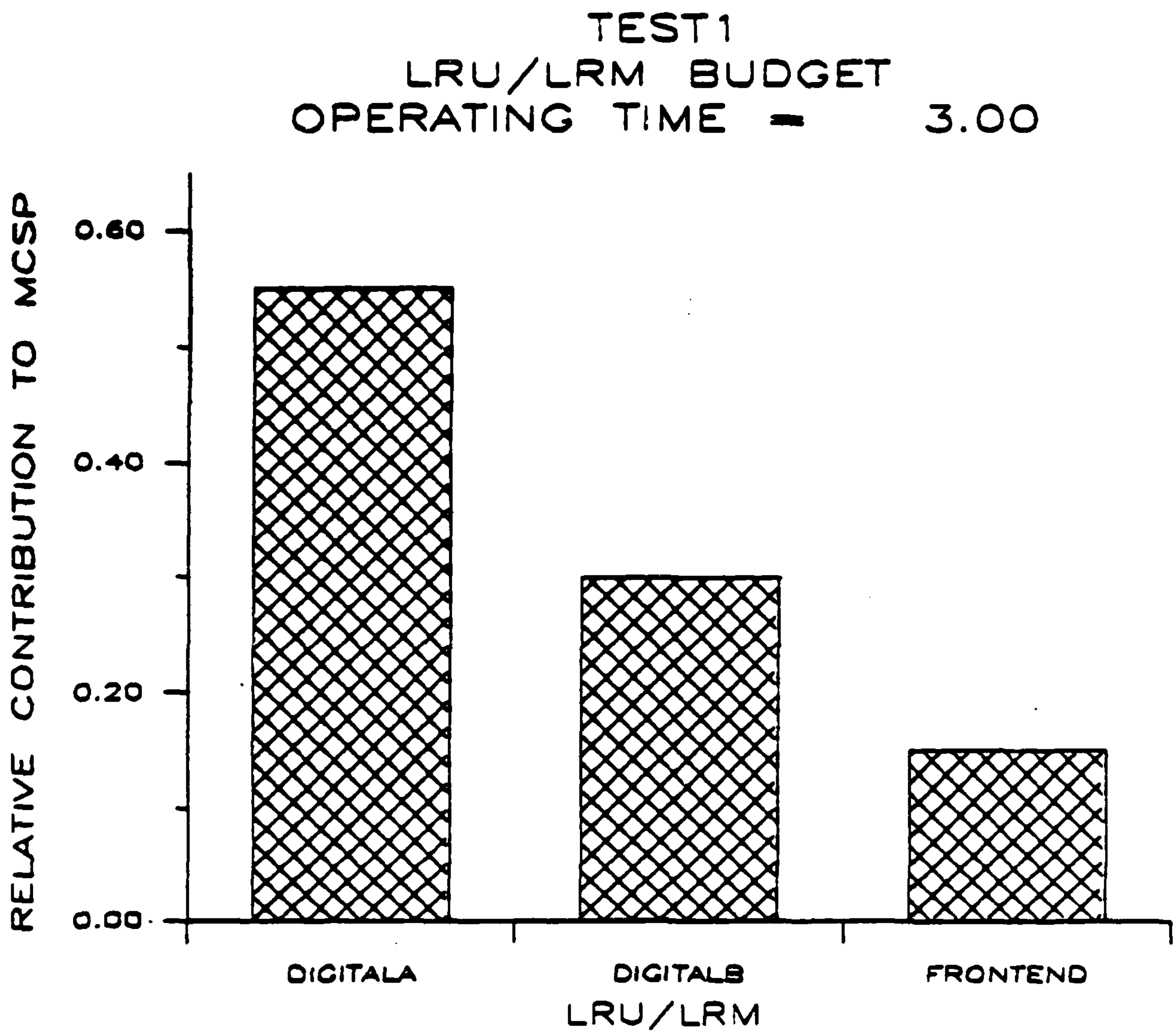


Figure C-6. Relative Contribution to MCSP versus LRU/LRM.

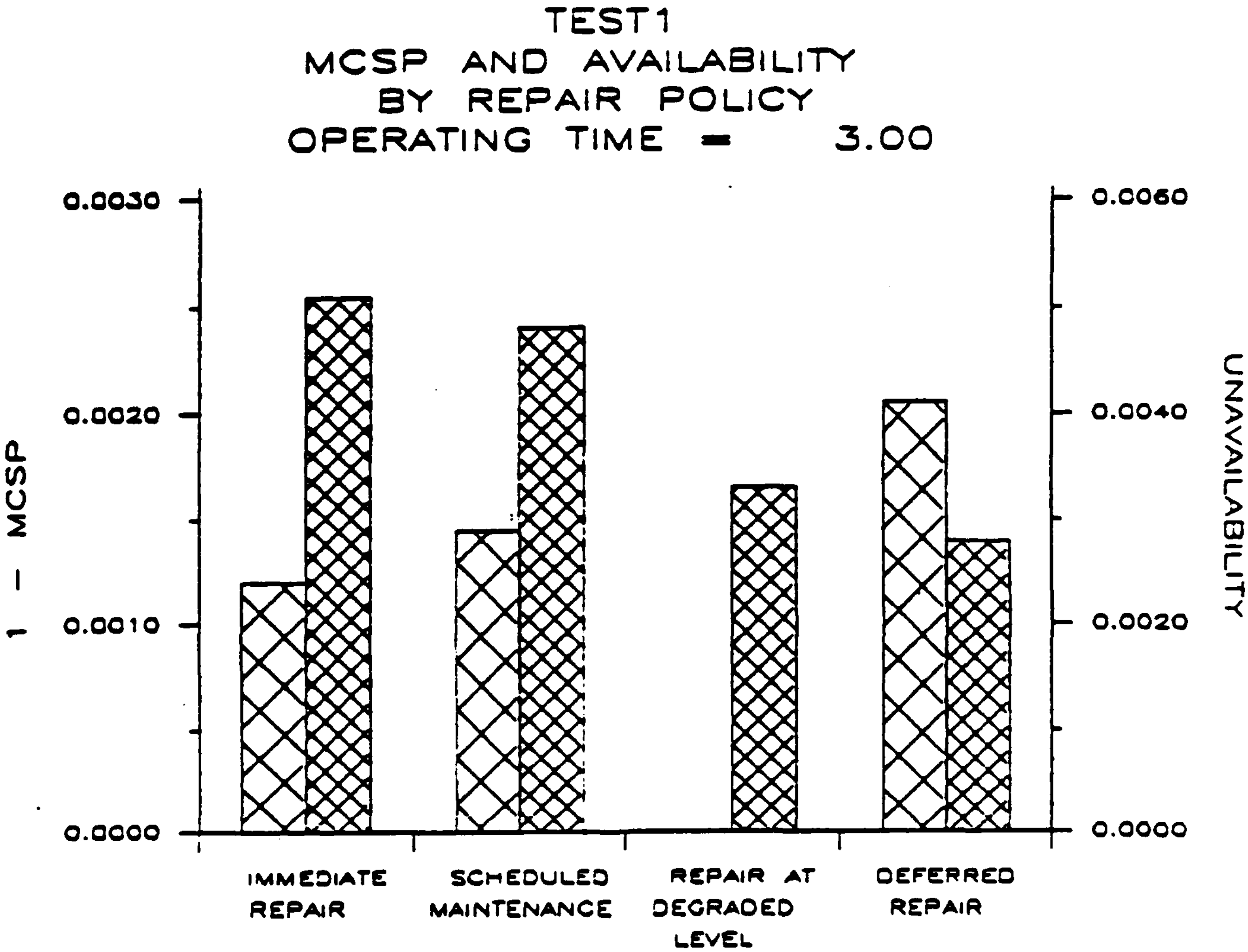


Figure C-7. MCSP and Availability versus Repair Policy.

END

1-87

DTIC